VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2012 Bc. Petr Meca

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Informační systém pro on-line rezervace

Information System for the on-line Reservations

2012 Bc. Petr Meca

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Petr Meca**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Informační systém pro on-line rezervace
Information System for the on-line Reservations

Zásady pro vypracování:

Cílem práce je rozšíření stávajícího rezervačních systému vytvořeného autorem diplomové práce. Rozšíření bude spočívat ve vytvoření dalších modulů a také v důkladém ověření fyzického návrhu databáze pomocí testovacího vytížení. Moduly, které budou v rámci práce vytvořeny jsou:

1. Statistiky rezervací (využití grafů jako na google analytics).
2. Synchronizace jednotlivých rezervací s google kalendářem uživatele.
3. Kreditní systém a využívání permanentek v rámci jednotlivých typů rezervací
4. Platební modul pro navyšování kreditu uživatele přes paypal a kreditní karty.
5. Vyvoření rozhraní pro externí autentizacii a autorizaci uživatele.

Dalšími kroky při řešení bude:
1. Sestavení state-of-the-art přístupů, které se používají pro autentizaci a autorizaci.
2. Rozšíření datové vrstvy rezervačních systému pro SQL Server.
3. Vytvoření testovacího vytížení a odladění fyzického návrhu databáze pro SQL Server.
4. Příprava auditu kritických databázových operací v SQL serveru.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Radim Bača, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2. května 2012 .....

Poděkování

Za ochotu, motivaci a trpělivost děkuji svému učiteli Ing. Radimovi Bačovi, Ph.D.

Abstrakt

Tato práce v první řadě zachycuje rozšíření mého online rezervačního systému o další moduly, a to konkrétně o permanentky, kreditový systém, statistiku, platební moduly a synchronizaci s google kalendářem. V druhé řadě se v práci objeví state-of-the-art přístupy pro autentizaci a autorizaci, rozšíření systému pro SQL Server, odladění fyzického návrhu databáze a audit databázových operací.

Klíčová slova: Rezervační systém, kreditový systém a permanentky, statistika, google kalendář, platební moduly, autentizace, SQL Server, audit

Abstract

First half of this thesis discusses my online reservation system extension, in which I added other modules, namely credit system and permanent passes, statistics, payment modules and a synchronisation with the google calendar. Second half of my thesis shows state-of-the-art methods of autentization and autorization, a system extension for the SQL server, a debug of the physical databasis concept and a database operations audit.

Keywords: Reservation system, credit system and permanent passes, statistics, google calendar, payment modules, autentization, SQL Server, audit

Seznam použitých zkratek a výrazů

PHP	–	Skriptovací jazyk běžící na straně serveru speciálně navržený pro potřeby webové stránky.
Zend Framework	–	Nádstavba nad jazykem PHP obsahující jisté přednastavené funkce.
SQL	–	Standardizovaný dotazovací jazyk používaný pro práci s daty v databázích.
MySQL	–	Databázový systém využívající jazyk SQL.
Power Plate	–	Typ přístroje pro zpevňování těla.
VacuShape	–	Typ přístroje na hubnutí těla.
Bloková rezervace	–	Typ rezervace, která se provádí po určitých blocích. Mějme otevírací dobu nějakého fitcentra, kterou rozdělíme do bloků (např. hodinových). Rezervace do těchto bloků se nazývá bloková rezervace.
Klientská část	–	Část rezervačního systému, kde klienti mohou operovat se svými rezervacemi apod.
Administrační část	–	Část rezervačního systému, kde má přístup majitel, případně zaměstnanci objektu, ve kterém se systém využívá, ti zde vidí veškeré rezervace všech klientů, jejich seznam atd. Odsud se ovládá celý rezervační systém.
Google analytics	–	Služba od společnosti Google pro sledování statistiky přístupů na internetové stránky.
AJAX	–	Asynchronní JavaScript, využívá se ve spojení například s PHP pro aktualizaci určité části internetové stránky, stránka se nemusí aktualizovat úplně celá.

Obsah

Seznam obrázků

1 Úvod

V této diplomové práci bych Vám rád představil a popsal můj vlastní informační systém nazvaný „FITSYSTEM“, který je koncipován pro online rezervace. Jeho vznik začal před pár lety, kdy jsem byl osloven, abych pro fitness centrum Hany Bany[1] vytvořil internetové stránky a zároveň také jednoduchý online rezervační systém. Této výzvy jsem se okamžitě chytil a započal práci na vývoji. Systém jsem začal programovat v jazyce PHP za pomocí Zend Frameworku, jako databáze byla použita MySql. Zpočátku stačilo, aby systém obsahoval pouze denní rozvrh, kalendář na přepínání dnů, seznam registrovaných klientů, rezervace přístrojů, blokování jednotlivých hodin v příslušném dni a blokování dnů. Rezervací přístrojů můžeme rozumět blokové rezervace jednotlivých zařízení jako Power Plate, Vacu Shape apod. Každý přístroj má v systému svůj vlastní sloupec rozdělen do zmiňovaných bloků. Je třeba ještě zmínit, že celý systém bylo třeba rozdělit do dvou částí, a to na klientskou část a administrační část, a to z důvodu jednoduchého ovládání systému. Pro obsluhu daného fitcentra to bohatě stačilo, nicméně s jeho rozvojem se postupně musel rozvíjet také rezervační systém.

Z důvodu neseriózního jednání některých zákazníků, se musel vytvořit modul „Penalizace prošlých rezervací “, prostřednictvím kterého byli jednotliví klienti v systému označování vykřičníkem, z důvodu že nepřišli na danou rezervaci a neomluvili se. Tato rezervace tedy blokovala možnost rezervování se jiným klientům na danou hodinu.

Dalším modulem, který bylo co nejříve třeba vytvořit byl modul „Archivace činnosti administrátorů“, respektive zaměstnanců fitness centra. Začaly totiž vznikat situace, kdy klientovi najednou zmizely rezervace, nebo naopak nějaké přibyly. Po prozkoumání databáze jsem následně zjistil, že tyto rezervace byly v administraci smazány, nebo naopak vytvořeny a bylo tedy třeba začít evidovat, kdo tyto operace provedl. Toto využití se potom osvědčilo i u brigádníků, kteří obsluhovali recepci a zároveň i rezervační systém.

Po určitém období se fitcentrum opět rozšiřovalo, a to nejen v rámci vlastního centra, ale rozšiřovalo se i do dalších měst. Během tří let se rozšířilo do 11 měst a v dnešní době už je v 15 městech. Tím začalo i další vylepšování systému, a to o nový typ rezervace. Jednalo se o rezervace na cvičení. Jednotlivá města totiž začala provozovat další služby, a to cvičení jako Zumba, Pilates apod. Tyto aktivity už nešlo rozdělit na příslušné bloky, tak jako přístroje, protože například Zumba se cvičila v příslušný den pouze jednou. Vytvořil se tedy nový rozvrh hodin - ten se nezobrazoval po dnech, tak jako u přístrojů, ale po týdnech. A to z toho důvodu, že jednotlivá cvičení se každý týden opakují, každý týden jsou stejně umístěna, a tak se nastavila kapacita daného cvičení (kolik klientů se může na danou aktivitu přihlásit). Vznikl tedy nový systém rezervací, a s tím i spojené výpisy těchto rezervací, blokace jednotlivých cvičení a archivace činností administrátorů v této oblasti.

Jelikož fitcentrum expandovalo na Slovensko, bylo nezbytné vytvořit jazykové mutace systému, bylo také nezbytné přepracovat zobrazování informací a vytvořit slovník,

[1] www.fithanybany.cz

pomocí kterého se daný systém překládal. Dále byl vytvořen modul „Náhradníci“, kdy se klienti mohli hlásit k jednotlivým přístrojům, či cvičením jako náhradníci a pokud se objevilo místo v rozvrhu, automaticky to daného klienta zarezervovalo a zaslalo mu email, že byl zarezervován na určitou hodinu a přístroj, či cvičení.

Poté se v systému dále vylepšovaly drobnosti jednotlivých hotových modulů, a to až do dnešní doby, kdy jsem si vlastní informační systém vybral jako diplomovou práci, ve které chci v první části popsat vytvoření dalších modulů jako permanenty pro jednotlivé typy rezervací, kreditní systém (veškeré rezervace budou odebírat kredity z účtu klienta), statistika (statistika veškerých rezervací, operací s kredity a vizualizace pomocí grafu podobného jako je v google analytics), synchronizace rezervací s google kalendářem, platební modul přes kreditní karty a vytvoření rozhraní pro externí autentizaci a autorizaci uživatele včetně sestavení state-of-the-art těchto přístupů.

V druhé části se dále budu specializovat na SQL Server, doposud totiž Fitsystém běží na MySQL databázi. Budu se tedy soustředit na rozšíření datové vrstvy systému na SQL Server, přípravu auditu kritických databázových operací a vytvoření testovacího vytížení a odladění fyzického návrhu databáze.

Praktickou část konkrétně administrační rozharní rezervačního systému si můžete vyzkoušet na adrese *http://diplomka.fitsystem.cz/admin*, kde přihlašovací údaje jsou:

jméno: diplomka
heslo: dip426lomka

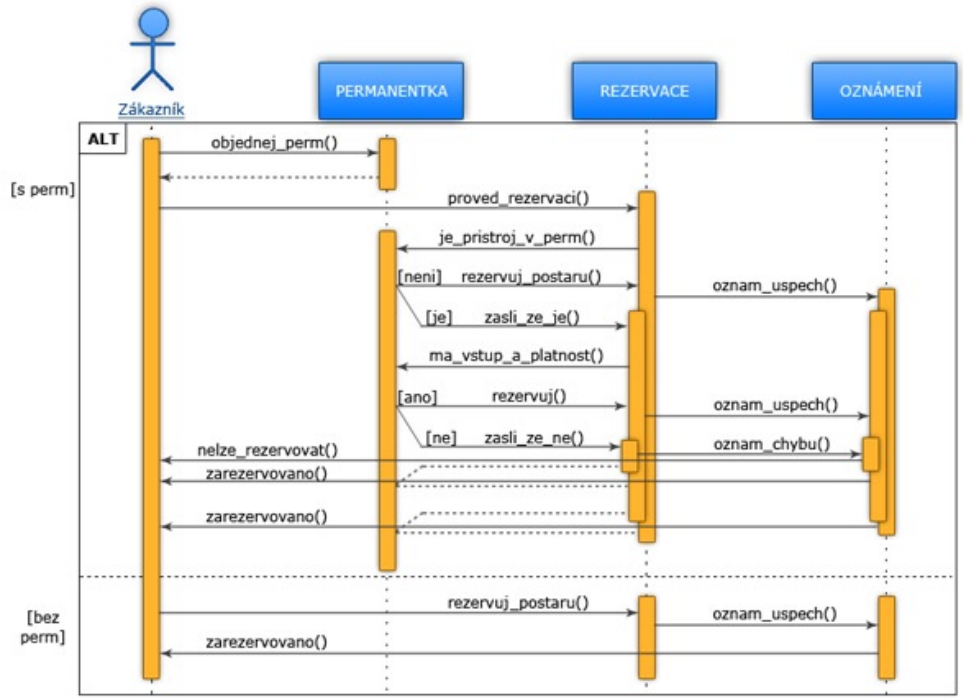
Klientskou část si můžete vyzkoušet na adrese *http://diplomka.fitsystem.cz*. Přístupy si můžete vytvořit v administrační části v záložce zákazník.

2 Permanentky

Velmi žádaným modulem byly permanentky, proto jsem je chtěl zahrnout i v této práci. Doposud systém fungoval tak, že se klienti mohli dopředu zarezervovat na jakýkoliv počet rezervací a po příchodu do fitcentra na recepci příslušnou rezervaci na daný den zaplatily. Fitcentrum ale po nějaké době začalo vytvářet akce, aby nalákalo další klienty a začalo vytvářet cenově výhodné vstupy, jako například 10 vstupů na přístroj Vacushape za sníženou cenu a díky tomu přišel na řadu modul permanentky.

2.1 Princip funkčnosti

Účelem permanentky je to, že si klienti mohou dopředu zarezervovat jen ten počet rezervací na daný přístroj či cvičení, který jim permanentka umožňuje. Na zbylé přístroje se mohou rezervovat starým systémem. Dejme tomu, že si klient objedná permanentku na přístroj Vacushape na 10 vstupů. Ve fitcentru ale mají navíc i přístroj Power Plate, tudíž se může rezervovat na oba dva přístroje, ale na Vacushape pouze 10 vstupů. Jakmile vyčerpá 10 vstupů, nemůže se na Vacushape zarezervovat a musí si dobít permanentku. Může si ji ale také nechat zrušit, čímž se mu opět povolí rezervace na daný přístroj, nicméně jeden vstup už nebude cenově výhodný.

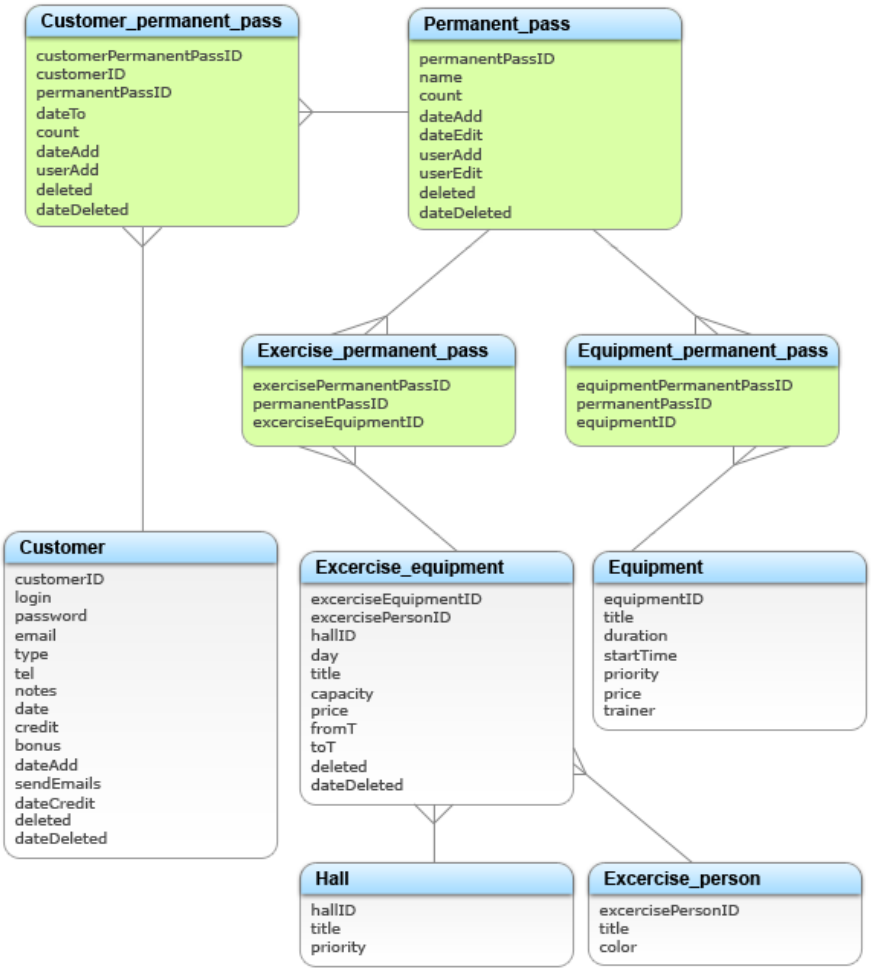


Obrázek 1: Sekvenčí diagram pro permanentky

Každá permanentka má také svou platnost, což znamená, že dané vstupy mohou klienti čerpat do určitého data. Ve většině případů fitcentra nastavují platnost jeden rok, je ale dobré mít v systému možnost zvolit pro permanentku jakékoliv období, protože je možné, že permanentky mohou fitcentra využívat na různé typy akcí, například letní slevy apod.

Princip funkčnosti permanentek můžete také vidět na obrázku (1)

2.2 Implementace



Obrázek 2: ER diagram tabulek pro permanentky

Prvním krokem k vytvoření rezervací za použití permanentek bylo zhotovení samotného vytváření dané virtuální permanentky. Po tvorbě ER diagramu(obrázek 2) a

analýze problému, co vše by se mělo v databázi ukládat do tabulky *permanent_pass* při vytvoření permanentky, jsem zvolil atributy, které můžete nalézt v příloze v datovém slovníku. V ER diagramu jsou nově vzniklé tabulky vyznačeny zelenou barvou.

Při vytváření permanentky je tedy třeba zadat její název, počet vstupů, a také které přístroje, či cvičení bude permanentka obsahovat. Ostatní atributy se automaticky doplní samy v závislosti na tom, kdo danou permanentku vkládal a v který časový okamžik. Atribut *deleted* a *dateDeleted* slouží pouze a jen pro informaci o tom, zda byla daná permanentka smazána či nikoliv, a to z důvodu, kdybychom někdy v budoucnu chtěli zpětně dohledat, co se s danou permanentkou stalo a kdy.

Pro uložení jednotlivých přístrojů a cvičení bylo třeba vytvořit další dvě tabulky, a to *equipment_permanent_pass* a *exercise_permanent_pass*. Obě dvě tabulky obsahují pouze ID daného přístroje, či cvičení a ID dané vytvořené permanentky. Tímto systémem si tedy vytvoříme seznam všech typů permanentek, které chceme v systému využívat.

Každý uživatel má v systému svou kartu, pomocí které může majitel fitcentra prostřednictvím administrační části změnit uživateli jeho údaje (jméno, heslo, datum narození apod.), může vidět jeho veškeré rezervace na cvičení a přístroje, archiv operací atd. Do této karty tedy přibylo i vložení permanentky danému klientovi. Při vkládání se jen vybere ze seznamu všech typů permanentek ta, kterou si klient koupil na recepci fitcentra a zároveň se zadá platnost, do kdy může permanentku využívat. Bylo tedy třeba vytvořit poslední tabulku pro ukládání těchto dat, a to tabulku *customer_permanent_pass*. Její atributy můžete opět nalézt v příloze v datovém slovníku.

Je ale možné, že do fitcentra přijde úplně nový klient, který ještě není v systému registrován, a proto je vkládání permanentky umístěno ještě na jedno místo, a to přímo při vytváření nového klienta. Toto slouží k urychlení práce recepčních ve fitcentru. Bylo by totiž zdlouhavé, kdyby recepční daného klienta zaregistrovala a až poté přešla do jeho karty a tam mu přidělila jednotlivé permanentky.

Na obrázku (2) můžeme vidět ER diagram znázorňující vazby mezi zmiňovanými tabulkami (tabulky se zeleným pozadím), kde atributy *customerID*, *equipmentID* a *exerciseEquipmentID* znázorňují postupně ID daného uživatele, ID přístroje a ID cvičení. Je tedy vidět, že jeden uživatel může mít více permanentek (permanentka na Vacushape, permanentka na Powerplate), a zároveň že jeden typ permanentky může být přidělen několika uživatelům. Co se týče přístrojů a cvičení, tak jedna permanentka může mít více cvičení a přístrojů, a zároveň jeden přístroj může být ve více typech permanentek.

Jakmile má tedy klient vloženou permanentku a provede rezervaci na určité cvičení, či přístroj, systém nejdříve zkontroluje, jestli opravdu permanentku má na dané aktivity. Pokud ano, zjistí, jestli má na ní dostatek vstupů a jestli datum platnosti permanentky je větší než datum dnešního dne. Jestliže ano, odečte se mu daná rezervace (aktualizuje se atribut *count* v tabulce *customer_permanent_pass* o -1) z permanentky a zároveň se v tabulce pro rezervace přístrojů či cvičení (podle typu permanentky) uloží k prováděné rezervaci ID řádku tabulky *customer_permanent_pass*. Důvodem je lepší rozeznání, které permanentce se musí zpět připočíst jeden vstup v případě, že by klient rezervaci smazal. Pokud na permanentce není dostatek vstupů nebo je permanentka prošlá, systém zahlásí,

že nemůže danou rezervaci provést. Pokud permanentku na příslušné přístroje či cvičení nemá, rezervace se provede běžným způsobem.

V systému se ještě vyskytují rezervace na tzv. kombinované přístroje. Kombinovaným přístrojem rozumíme například jednu rezervaci na Vacushape + Power Plate (tato kombinace stojí méně než kdyby si klient zarezervoval zvlášť Vacushape a potom Power Plate). Každý přístroj v systému má svůj vlastní sloupec rodělený do časových bloků (například půlhodinových), do kterých se provádí rezervace. Když si tedy v systému chcete zarezervovat kombinaci dvou přístrojů, po kliku do příslušného bloku jednoho přístroje vyskočí okno s otázkou, jestli opravdu chcete tuto rezervaci uskutečnit a zároveň se pod tímto dotazem objeví seznam přístrojů, které je možné s původním přístrojem zkombinovat (kombinace se nastavují v databázi příslušné tabulky). Pokud zvolíte druhý přístroj ke kombinaci, systém tedy provede dvě rezervace (musí zablokovat oba dva přístroje), ale musí odebrat jen jeden vstup z permanentky. Na tuto operaci tedy bylo nutné dané permanentky připravit.

Pro přehlednost mají uživatelé ve své klientské části seznam veškerých svých permanentek a k nim přiděleny počty vstupů, aby viděli, kolik rezervací si mohou ještě zarezervovat. Nemohou si ale dané permanentky sami upravovat či mazat, tyto operace se mohou provádět jen na recepci v administrační sekci systému.

3 Kreditový systém

Dalším modulem, který jsem chtěl v rámci diplomové práce vytořit, je kreditový systém. Vznikl na základě mé vlastní iniciativy, nebyl vyžadován žádným fitcentrem, chtěl jsem pouze potenciálním zájemcům o rezervační systém nabídnout další, úplně jinou formu využívání systému a jeho rezervací.

3.1 Princip funkčnosti

Jeho princip spočívá v tom, že daný klient má v systému svůj účet, na kterém má určitý počet kreditů. Tyto kredity získá buď tím, že si je zakoupí na recepci fitcentra většinou v poměru 1:1 vůči reálným penězům, anebo přes platební bránu PayU (viz kapitola 4). Další možností je ještě získání kreditu v rámci přidělení bonusů. Pokud klient nevyužije platební bránu, je mu kredit vkládán v jeho osobní kartě v administrační části sytému.

Kredit lze využít k rezervaci přístrojů či cvičení, ale také třeba k nákupu různých produktů, jako například energy drink, anticelulitidový krém apod., které fitcentrum nabízí na recepci. Tento obchod si popíšeme níže v kapitole 3.3.

Pokud klient začne provádět určité rezervace na přístroje či cvičení, při každé této rezervaci se mu z účtu odečte tolik kreditů, kolik dané cvičení či přístroj stojí. Je třeba ale předem tyto ceny jednotlivých aktivit (přístroje, cvičení) v systému nastavit v databázi v tabulkách reprezerntujících jednotlivé přístroje a cvičení. Většinou se ceny za danou aktivitu nastavují opět v poměru 1:1 vůči reálným pěnězům, takže když například jeden vstup na Power Plate stál 120 Kč, v systému bude stát 120 kreditů.

Při každé provedené rezervaci je nutné uchovávat v databázi u každé rezervace také kolik odebrala kreditů a kolik bonusů. Uchování je prováděno z toho důvodu, kdyby klient smazal danou rezervaci, aby mu systém vrátil zpět počet kreditů a bonusů, které tato rezervace stála. Navíc je toto uchovávání také užitečné v tom, že dejme tomu přístroj Power Plate stojí 110 kreditů. Klient si zarezervoval dopředu 5 rezervací, tudíž u každé z těchto rezervací je zaznamenáno 110 kreditů. Za 5 dní se fitcentrum rozhodne, že přístroj Power Plate zdraží na 150 kreditů. Klient se navíc rozhodne určité rezervace smazat, ale systém mu vrátí ten počet kreditů, kolik přístroj stál v té době, než se zdražil, což je správně. Kdybychom tato data u rezervací neuchovávali, klientovi by při smazání přišlo 150 kreditů a ještě by na tom vydělal.

U zavedení kreditového systému bylo nutné také pamatovat na kombinované přístroje, tak jako v kapitole 2. Cena kombinovaných přístrojů je nižší, než když byste si tyto přístroje zarezervovali jednotlivě, a proto bylo třeba nastavit jednotlivým kombinacím, kolik kreditů budou při rezervací odebírat. Tyto hodnoty se tedy nastavily v databázi přímo v tabulce příslušných kombinací.

Stejně jako tomu bylo u permanentek, i daný kredit má svou platnost. Většinou je defaultně nastavena na jeden rok, je ale možné ji nastavit na jakoukoliv dobu, kdyby

si fitcentrum například chtělo vytvářet akce na kredity na určité období (například na měsíc).

Kredity lze v kartě každého klienta také odebrat. Například recepční přidá nesprávný počet kreditů, splete se, a proto musí existovat možnost odebrání určitého množství kreditů. Samozřejmě, že by šlo k odebrání využít jejich přidání, akorát by se přidávala záporná hodnota, nicméně pro přehlednost jsem vytořil druhou kolonku pro odebrání kreditů. V systému tedy lze přidat kredit, přidat bonus a jako druhá možnost odebrat kredit a odebrat bonus.

3.2 Bonusy

S vytvořením krediového systému vznikl nápad na vytvoření bonusů, které za určitých podmínek navyšují celkový počet kreditů. Tyto bonusy se klientům přiděluji v každém fitcentru jinak. Záleží jaký systém s kredity si dané fitcentrum samo vytvoří. Jedná se o to, že například do fitcentra přijde klient a nechá si nahrát 1000 kreditů za které zaplatí 1000 korun. Druhý klient si objedná 5000 kreditů, zaplatí za ně 5000 korun, ale dostane k tomu navíc 500 kreditů jako bonus, že si koupil velké množství kreditů. Takže celkem má 5500 kreditů a může si zarezervovat o něco více rezervací na přístroje nebo cvičení.

V systému ale tyto bonusy eviduji zvlášť, nijak je nemíchám s kredity dohromady, pouze v určitých výpisech, ale v databázi mají každý svou kolonku, a to z toho důvodu, at' klienti ve své klientské části v archivu operací vidí, že opravdu bonus dostali. Pokud by navíc chtěli pomocí kreditu nakoupit nějaké produkty (energy dring, kafe apod.), tak tyto položky nelze odebírat z bonusů, protože by tím „ničily“ cenu daného produktu, za tyto položky je třeba platit kreditem.

Dále počet kreditů vlastně reprezentuje počet peněz, které klienti utratili ve fitcentru. Tato data se potom zobrazují ve statistice a pokud by byly kredity smíchány s bonusem, nebylo by možné poznat, kolik peněz se za určité období ve fitcentru na kreditech vydělalo. O statistice budu ale psát více v kapitole 5.

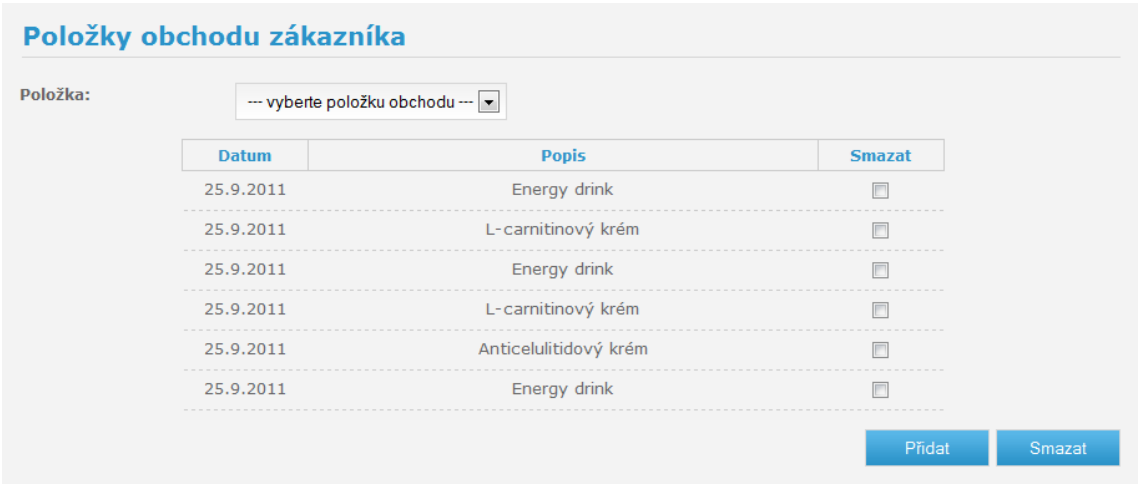
Princip bonusu tedy hlavně spočívá v tom, že je klientům dopřáno připsání většího počtu kreditů než si objednali, čímž si můžou zarezervovat více rezervací.

3.3 Obchod s produkty

Jak už jsem zmínil v kapitole ohledně bonusů, v systému existuje jednoduchý obchod pro nákup položek, jako například *Energy drink*,*Mysli tyčinka*, *kafe* apod., tyto položky se prodávají na recepci a nakupují se za reálné peníze. Díky kreditovému sytému má klient vlastní účet s určitým množstvím kreditů a pokud jich má dostatek, může si za ně nakoupit příslušné produkty. Nákup se opět provádí na recepci, kde recepční v kartě daného klienta provede nákup dané položky a klientovi se odečte kredit. Čímž nemusí nic platit hotově

a nemusí si tím pádem ani nosit peněženku do fitcentra. Vložení nějaké položky klientovi můžeme vidět na obrázku (3)

Každá položka obchodu tedy musí obsahovat počet kreditů, který se klientce odečte při nákupu. Pro tyto položky existuje tabulka *shop*. Atributy této tabulky, můžete opět najít v příloze v datovém slovníku. Jednotlivé položky této tabulky tedy reprezentují jednoduchý sklad zásob. Při nákupu produktu se v kartě klienta ještě musí evidovat to, které produkty mu byly prodány. Tato data uchovává tabulka *shop_reservation* a díky této tabulce se potom také aktualizuje statistika v rámci toho, kolik se prodalo produktů, kolik to stálo kreditů apod. Navíc klienti potom ve své klientské části vidí v archivu operací, že jim byl ten a ten den objednán daný produkt, a že stál určitý počet kreditů.



Obrázek 3: Položky obchodu u zákazníka

3.4 Archiv činnosti administrátorů

Tento modul funguje tak, že pokud jakýkoliv administrátor v systému provede příslušné operace s rezervacemi, evidují se tyto operace v kartě klieta. Údaje archivu jsou reprezentovány formou tabulky viz. obrázek (4). Evidují se vložené i smazané rezervace, úpravy na kartě klienta a navíc se zde nenachází nejen činnost administrátorů, ale také činnost daného klienta.

Kreditový systém do tohoto archivu zasáhl v tom smyslu, že se začalo evidovat více položek, a to vložené kredity a bonusy klientovi, odebrané kredity a bonusy klientovi a aktuální zůstatek kreditů na účtě. Dále kolonky pro odebrané a vložené kredity slouží nejen pro evidenci klientských kreditů, ale pokud řádek v archivu reprezentuje vloženou rezervaci, kolonka odebraných kreditů reprezentuje v tomto případě, kolik daná rezervace odbrala kreditů z účtu klienta. Analogicky potom i u smazané rezervace a přidaných kreditů.

Archív operací

datum	operátor	vložil rezervaci na den	smazal rezervaci na den	na přístroj (cvičení)	vložil (vrátil) kredit	vložil (vrátil) bonus	odebral kredit	odebral bonus	upravil	zůstatek vč. bonusů
27.3.2012	klientka	27.3.2012 08:10		Vacu Shape I.	0	0	169	0		9981118
24.3.2012	admin	24.3.2012 08:50		Vacu Shape II.	0	0	169	0		9981287
24.3.2012	klientka	24.3.2012 06:50		Vacu Shape II.	0	0	169	0		9981456
24.3.2012	klientka	24.3.2012 10:10		Power Plate II.	0	0	215	0		9981625
24.3.2012	klientka	24.3.2012 09:30		Power Plate II.	0	0	215	0		9981840
24.3.2012	klientka	24.3.2012 08:50		Power Plate II.	0	0	215	0		9982055
24.3.2012	klientka	24.3.2012 08:10		Power Plate II.	0	0	215	0		9982270
24.3.2012	klientka	24.3.2012 07:30		Power Plate II.	0	0	215	0		9982485
24.3.2012	klientka	24.3.2012 06:50		Power Plate II.	0	0	173	42		9982700
24.3.2012	klientka		24.3.2012 06:50	Power Plate I.	0	0	0	0		9982915
24.3.2012	klientka	24.3.2012 06:50		Power Plate I.	0	0	0	0		9982915
24.3.2012	klientka		24.3.2012 07:30	Power Plate I.	0	0	0	0		9982915
24.3.2012	klientka		24.3.2012 07:30	Vacu Shape II.	0	0	0	0		9982915
24.3.2012	klientka		24.3.2012 06:50	Power Plate II.	0	0	0	0		9982915
24.3.2012	klientka	24.3.2012 07:30		Power Plate I.	0	0	0	0		9982915

1 2 3 4 5 Další > Poslední >>

Obrázek 4: Archiv operací

3.5 Náhradníci

Jak už jsem psal v úvodu, v systému existuje modul náhradníci, který slouží k tomu, aby se klienti nahlásili jako náhradník na příslušné cvičení, či přístroj v určitém časovém intervalu. Pokud se do té doby, než cvičení začne, uvolní nějaké místo, klient je automaticky zarezervován a je mu zaslán email o tom, že má danou rezervaci.

Při zavedení kreditového systému bylo třeba aktualizovat i tento modul, a to v tom smyslu, že je nezbytné při automatické rezervaci odečíst i příslušný počet kreditů. Zde ale nastává problém, a to ten, že když je klientovi zaslán email, nemusí si ho vůbec přečíst a nemusí na dané cvičení přijít. Navíc když se nedostaví, přijde o určité peníze, protože mu automatická rezervace odebrala kredit.

Pro tento modul se nabízí možnost využití zaslání SMS klientovi pro upozornění, že se mu zarezervovalo cvičení, nicméně když jsem tuto variantu probíral s různými fitcentry, nikdo ji nechtěl z toho důvodu, že by za SMS zprávy hodně platili. Proto se někteří rozhodli, že daný modul v rámci kreditového sytému nebudou vůbec využívat. Některým ale tento modul přesto vyhovoval a rozhodli se, že jej využívat budou a pokud se stane situace, že klient o rezervaci vůbec nebude vědět, vrátí mu kredit zpět a budou k

ní tímto vstřícní. Z tohoto důvodu se vždy každého majitele při zakládání rezervačního sytému ptám, jestli daný modul chtějí nebo ne a vysvětluji, jaké to může mít důsledky.

3.6 Nepotřebné moduly při využívání kreditového systému

Jestliže se majitel fitcentra rozhodne zavést kreditní systém, nebude nutné využívat některé ze stávajích modulů, a to například modul nevyužitých rezervací. Tento modul totiž slouží k označení klienta, který na danou rezervaci nepřišel a pokud přijde příště, je za tuto prošlou rezervaci finančně penalizován. Pokud ale v systému bude nastaven kreditní systém, tyto penalizace nebude nutné využívat, a to z toho důvodu, že pokud si klient zarezervuje nějaký přístroj či cvičení, okamžitě se mu po provedení rezervace odečte daná částka za příslušnou aktivitu a pokud se rozhodne nepřijít, kredit mu propadne. Fitcentrum má vlastně toto cvičení jako kdyby zaplacené a nevadí, že daný přístroj nebo cvičení je touto rezervací blokovaný. V předchozím případě toto problém byl, protože se klienti na daný přístroj nemohli zarezervovat a fitcentrum tím prodělávalo, protože rezervaci nemělo ještě zaplacenou.

Další modul, který není možné využívat v rámci kreditového systému jsou již zmiňované permanentky z minulé kapitoly. Permanentky totiž obsahují určitý počet vstupů a nijak nesouvisí s kredity. Jejich napodobeninu ale kreditový systém zaujímá v tom, že lze klientovi nabít určitý počet kreditů, který je například desetinásobkem ceny příslušného přístroje nebo cvičení. Tím určíme, že klient může provést 10 rezervací na danou aktivitu. Problém ale nastává v tom, že klient má přístup ke všem těmto cvičením nebo přístrojům, proto je ještě možnost mu zpřístupnit pouze ty přístroje, na které si koupil kredit. Není potom možné se na nic jiného zarezervovat.

4 Platební moduly

Klienti si do nedávna mohli nabíjet kredity pouze ve fitcentru na recepci. S modulem **platební moduly** klientům rozšířím danou možnost o pohodlnější a příjemnější verzi, budou si moci v klidu z domova dobít kredit a ihned provádět rezervce.

Existují dva způsoby jak tento modul implementovat do systému, a to

- naprogramovat si ho sám s využitím technologie banky, u které by mělo fitcentrum založený účet. Například pokud by měli účet u České spořitelny, musel bych rezervační systém napojit na jejich **3D Secure** systém [2].
- využít už hotové platební moduly se všemi možnými typy plateb jako jsou například PayPal, Moneybookers, PayU, GoPay, či PaySec.

V mém případě jsem si pro implementaci vybral druhý způsob, a to z toho důvodu, že když už je podobná služba vyvinutá, mají její vývojaři již odladěné veškeré chyby a funguje jim to, tak proč ji nevyužít než se trápit s vývojem vlastních věcí a odlaďovat vzniklé chyby a navíc brát na sebe zodpovědnost za přenos peněz. Navíc zmiňované služby již mají v sobě zabudováno několik typů plateb, takže se klient může rozhodnout jak peníze zaslat a využít tak například služeb banky, u které má účet a poslat tak peníze během několika sekund. Dále jsem si druhou variantu vybral z toho důvodu, že například u už zmiňované České spořitelny musíte za využívání jejich služeb zaplatit aktivační poplatek, který není nijak malý a potom vám ještě z každé platby odebírají určitou částku jako další poplatek [9]. Takže vlastní implementace by byla i finančně náročnější.

Když jsem se tedy rozhodl využít druhou možnost, bylo třeba se rozhodnout pro určitý typ služby. Všechny tyto služby pracují na podobném principu, a to formou tzv. **elektronické peněženky**, což je vlastně virtuální účet na internetu, kde si můžete vkládat peníze, vybírat a zasílat je dalším lidem, kteří danou službu využívají, a to během několika sekund. Za aktivaci této služby nic neplatíte, platí se pouze poplatky za převod peněz z jednoho účtu na druhý. Dále každou tuto službu můžete využívat ve svém e-shopu či na jiné stránce, kde potřebujete, aby lidé mohli platit kreditními kartami. Tyto platby potom chodí na váš virtuální účet a odtamtud si je můžete zasílat na svůj bankovní účet, opět za určitý poplatek. Tyto popatky nejsou stejné pro každého uživatele, vždy se vypočítavají na základě služby kterou provozujete, na základě obratu apod.

Mezi nejznámější a nejdéle fungující službu patří PayPal, která je celosvětová a využívá k platbě několik druhů kreditních karet. Bohužel ale neobsahuje další typy plateb jako například „rychlé platby“ od různých typů bank. Například Komerční banka má rychlou platbu nazvanou **Mojeplatba**, díky které mohou klienti této banky provádět transakce během několika sekund. Tyto rychlé platby mají i další banky a služby které tyto platby podporují jsou PayU a GoPay. Je to tím, že se více zaměřují na Českou republiku. GoPay je přímo česká firma a vlastníkem PayU je Aukro, které si pro svůj web samo vyvinulo

[2] http://www.csas.cz/banka/content/inet/internet/cs/sc_1585.xml

tuto platební metodu. Jelikož obě dvě služby nabízejí stejné možnosti plateb, vybral jsem si tu, která se využívá na tak velkém portále jako je Aukro, a proto jsem zvolil službu PayU.

4.1 Služba PayU

Předem bych chtěl uvést, že pro provoz této služby je nutné mít podepsanou smlouvu s PayU, do té doby vám nezašlou přihlašovací údaje a potřebné kódy pro komunikaci s jejich API. Během zpracovávání této diplomové práce si rezervační systém objednalo jedno plzeňské fitcentrum Goldenbody [3], které dané platby chtělo, a proto jsem tuto firmu využil k naimplementování PayU do mého systému. V praktické části diplomové práce se tedy tento modul nevyskytuje z důvodu už zmiňované smlouvy, ale pro představu, jak platební modul vypadá, se můžete zdarma zaregistrovat v plzeňksém fitcentru [4]. Popíšeme si tedy jak daný modul funguje.

4.1.1 Založení služby

K tomu abych mohl mít službu PayU implementovanou v rezervačním systému, bylo třeba provést tyto kroky:

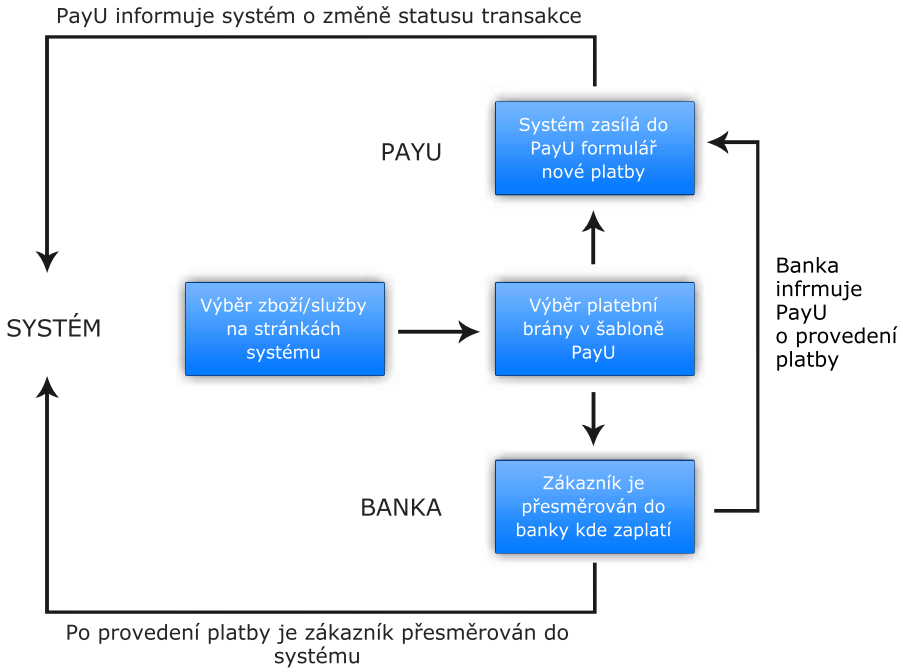
- Zaslat PayU žádost prostřednictvím internetového formuláře, kde je třeba vyplnit název společnosti, která danou službu bude využívat, IČ, DIČ, adresu společnosti a URL adresu na které bude služba fungovat. V dalším kroku je ještě nutné zadat číslo bankovního účtu, se kterým bude účet na PayU propojen.
- Po odeslání formuláře a ověření údajů nám PayU do 48 hodin zašle návrh smlouvy a nabídku kolik daná služba bude stát, kolik budou poplatky za transakce na účtu apod.
- Jakmile je smlouva podepsaná a doručena PayU, zašlou nám přihlašovací údaje do jejich administrační části a tam už si sami nastavíme propojení služby PayU s našim systémem. Navíc v této administraci vidíme seznam všech plateb, jejich správu a možnost potrvrzovat jednotlivé transakce.
- Je ale ještě nutné v našem systému naimplementovat dané propojení. K tomu abychom zjistili, že platby opravdu fungují, PayU nabízí možnost testovacích plateb, kde si přímo můžete nastavit, jestli se platba provedla nebo ne, jestli je dokončená nebo čeká na vyřízení apod.
- Jakmile máme implementaci dokončenou, zašleme o tom informaci PayU a oni nám zpřístupní veškeré další platební kanály a testovací platby budou zrušeny.
- Po zveřejnění platebních kanálů je platební systém připraven k použití.

[3] http://www.goldenbody.cz
[4] http://goldenbody.fitsystem.cz

4.1.2 Implementace a propojení PayU s Fitsystémem

Služba PayU nám po úspěšné registraci zašle **POS_ID** a **POS_AUTH_ID**, což jsou hodnoty, které se při každé platbě prostřednictvím rezervačního systému posílají PayU, aby šlo rozpoznat, jestli jsme to opravdu my. Dále se zasílá **SESSION_ID**, což je jednoznačné určení dané platby, aby se člověk potom na ni mohl odkazovat, případně zjišťovat, v jakém stavu daná platba je. Dále do systému bylo třeba naimplementovat výpis veškerých platebních kanálu a po výběru jednoho z nich uživatele přesměrovat na PayU. Tento výpis je prováděn pomocí javascriptu, který načítá všechny platební kanály ze stránek PayU.

Princip funkčnosti můžeme vidět na obrázku (5)



Obrázek 5: Princip funkčnosti PayU

Nejdéle ale trvalo naimplementovat odchytávání veškerých chybových upozornění, bylo třeba zajistit veškeré možné problémy, které mohly nastat a vypisovat tak různé statusy od PayU. Celkově tato implementace PayU do Fitsystému zabrala asi 3 dny, včetně testování plateb. Z toho důvodů si také podrobnou implementaci dané služby můžete stáhnout z oficiálnch stránek PayU[5].

Nakonec ještě bylo třeba v databázi rezervačního systému evidovat veškeré transakce a chybová upozornění, aby klienti, kteří službu využijí, měli přehled o svých transakcích a

[5]http://www.payu.cz/ke-stazeni

chybách, které provedli. Tabulky pro tuto evidenci jsou *payu_payment* a *payu_info*. Jednotlivé atributy těchto tabulek jsou v datovém slovníku v příloze. V administrační části jsem toto nemusel implementovat, protože majitel fitcentra se na tyto výpisy může podívat přímo v administraci PayU.

4.1.3 Platební kanály

Jak už jsem psal výše, PayU obsahuje několik typů platebních kanálů pro rychlé převody peněz. Jejich seznam můžete vidět na obrázku (6).

Uživatel, který si chce tedy v rezervačním systému nabít kredit, si zvolí ze seznamu banku, u které má účet a provede rychlou platbu. Pokud se jeho banka v seznamu nenachází, může využít klasický bankovní převod, nicméně tento převod může trvat i déle než 24 hodin.

rychlý převod z Komerční banky	KB	Mojeplatba	zpracování v reálném čase
rychlý převod z GE Money bank	GE Money Bank	GE Money bank	zpracování v reálném čase
rychlý převod z mBank	mBank	mPeníze	zpracování v reálném čase
rychlý převod z Raiffeisen banky	Raiffeisen BANK	e Platby	zpracování v reálném čase
rychlý převod z Volksbank	VOLKSBANK	Volksbank	zpracování v reálném čase
rychlý převod z Fio banky	Fio banka	Fio banka	zpracování v reálném čase
platební karty	MasterCard VISA	VISA, MasterCard	zpracování v reálném čase, bypass přes ewallet
běžný bankovní převod		pro ostatní bankovní ústavy	při odeslání platby do 15:00 prostředky připsány v reálném čase následující den
hotovostní platba na terminálech Sazka a přepážkách České pošty	SUPERCASH	superCASH	zpracování v reálném čase
platba poštovní poukázkou		složenka	poukázaná peněžní částka je vyplacena zpravidla do tří pracovních dnů ode dne podání

Obrázek 6: Typy platebních kanálů

4.1.4 Průběh platby

Pro lepší představu můžete průběh platby vidět na obrázku (7).

Obrázek 7: Průběh platby u PayU

Funguje to tedy následovně.

- Zákazník se rozhodne zakoupit předmět a zvolí banku, ve které chce provést platbu (banka, ve které má účet).
- Následně jej systém přesměruje na formulář, jehož pomocí se přihlásí do svého bankovního účtu.
- Formulář pro zadání platby je připravený s konkrétními údaji, které banka obdržela od systému PayU (číslo účtu, variabilní symbol, částka).
- Zákazník potvrdí platbu a ta je následně převedena prostřednictvím platebního kanálu na účet určený pro službu PayU u této banky.
- Zákazníci mohou také využít možnost platby kreditní kartou (přes ewallet) nebo standardním bankovním převodem.
- Nyní můžete platbu od zákazníka přijmout či odmítnout. Tuto částku si můžete nechat kdykoliv vyplatit z Vašeho účtu v systému PayU na Váš bankovní účet.

5 Statistika rezervací a kreditů

Tento modul jsem si sám vybral jako součást diplomové práce, protože si myslím, že je velice důležitý a nezbytný pro jakékoliv fitcentrum či jiný objekt. Jeho význam spočívá v tom, že si majitelé fitcentra mohou zjistit, kolik rezervací bylo například za minulý měsíc, případně kolik jich bylo u konkrétních přístrojů, některé přístroje mohou také zrušit, pokud obsahovaly minimální počet rezervací, či naopak přidat další přístroj. Dále si mohou například zjistit kolik bylo vloženo klientům kreditů za určité období, čímž se dozví, kolik si za dané období vydělali peněz apod.

5.1 Rozdělení statistiky

Statistiku si vždy můžete zobrazit na určité období. Pokud vstoupíte v systému do záložky „statistika“, objeví se vám data za celý dnešní den. Poté si můžete zvolit různé časové rozpětí, ve kterém chcete vypsat statistiku tak, jak je vidět na obrázku (8). Statistika se zobrazuje jak klientům v klientské části systému, tak v administrační části majitelům fitcentra. Rozdíl mezi těmito zobrazeními je v tom, že v klientské části klient vidí pouze a jen svá data, nevidí žádná cizí. V administrační části má majitel možnost sledovat statistiku celého svého fitcentra a zároveň také konkrétně jednotlivé klienty.

Obrázek 8: statistika celkových počtů

Jak už jsem naznačil v předchozím odstavci, statistika je rozdělena do dvou míst, a to do administrační části a klientské části. Obě tyto verze se dále dělí na

- **statistika celkových počtů** - jedná se o data, která zobrazují souhrnné (celkové) počty za určité zvolené období, zobrazují statistiku celého fitcentra. Tento typ můžeme vidět na obrázku (8). Dále nabízí možnost po najetí myší na jeho konkrétní hodnoty zobrazit detailnější statistiku, a to pro konkrétní uživatele. Například po najetí na hodnotu reprezentující celkový počet rezervací se nám zobrazí tabulka s jednotlivými klienty, kteří tyto rezervace provedli.
- **statistika konkrétních přístrojů a cvičení** - tato data reprezentují konkrétní hodnoty rezervací nebo kreditů pro jednotlivé přístroje, cvičení či produkty v obchodě. Součet

hodnot těchto jednotlivých statistik potom musí odpovídat souhrným počtům z předchozího bodu. Tento typ statistiky můžeme vidět na obrázku (9).

Přístroje	Počet rezervací	Smazaných rezervací	Aktuálních rezervací	Vytíženost přístroje	Odebraných kreditů (vč. bonusů a vrácených kr.)
Vacu Shape I.	15	9	6	1%	1014
Vacu Shape II.	7	2	5	1%	1014
Power Plate I.	3	1	2	0%	430
Power Plate II.	2	0	2	0%	430
Pneuven	0	0	0	0%	0
Sauna	0	0	0	0%	0
Tenis	7	0	7	1%	315
Solárium	1	1	0	0%	0
Badminton	2	1	1	0%	55
Squash	4	1	3	1%	165
Bowling	0	0	0	0%	0
Nohejbal	1	0	1	0%	50
Dráha 1	4	0	4	0%	200
Dráha 2	1	0	1	0%	50
Dráha 3	1	1	0	0%	0
Lucie	4	1	3	1%	150
Vlaďka	4	1	3	1%	150
Petra	1	0	1	0%	50
Celkem	**57**	**18**	**39**		**4073**

Obrázek 9: statistika konkrétních přístrojů

Každý z těchto typů se dále dělí na

- **statistika rezervací** - jedná se o data zobrazující statistiku jednotlivých rezervací na přístroje, cvičení, či statistiku objednaných produktů z obchodu. Také se u jednotlivých přístrojů uvádí jejich vytíženost v procentech.
- **statistika kreditů** - statistika kreditových operací. Kolik bylo daným přístrojem odebráno kreditů, kolik kreditů bylo vloženo, či odebráno klientovi, kolik kreditů odebral určitý produkt v obchodě apod.

Nyní si podrobněji rozebereme jednotlivé typy statistiky.

5.1.1 Statistika celkových počtů rezervací

Tato první část, kterou můžeme vidět v levé části obrázku (8), se zabývá pouze celkovým počtem provedených rezervací v systému za určité zvolené období. Zde je uvedeno:

- **vložených rezervací** - jedná se o veškerý počet všech rezervací na přístroje, cvičení a objednané produkty v obchodě za zvolené období. V této hodnotě jsou započítány i rezervace, které v daném období byly smazány.
- **smazaných** - tato hodnota, jak už její název napovídá, reprezentuje počet smazaných rezervací jak přístrojů a cvičení, tak i z obchodu.
- **aktuálních** - hodnota určující pouze rozdíl předchozích dvou hodnot, určuje aktuální počet rezervací v systému za zvolené období.

5.1.2 Statistika celkových počtů kreditů

Zde vysvětlím druhou část (druhý a třetí sloupec) z obrázku (8), která je oproti první poněkud obsáhlejší. Jedná se o kreditové operace, a to jak v rámci uživatelů, tak v rámci jednotlivých rezervací. První ze dvou sloupců obsahuje operace v rámci klientů, a to konkrétně:

- **vloženo do kreditů** - jedná se o počet vložených kreditů všem daným klientům v určitém časovém období. Jsou to vlastně nakoupené kredity, takže toto číslo také uvádí, kolik peněz lidé utratili ve fitcentru za kredity. Samozřejmě předpokládáme, že má fitcentrum nastaveno, že jeden kredit se rovná jedné koruně.
- **do bonusů** - tato hodnota určuje kolik se všem daným klientům vložilo bonusů.
- **odebraných** - počet odebraných kreditů a bonusů dohromady. Ve fitcentru se může stát, že někdo omylem klientovi nabije více kreditů než měl, a proto je musí klientovi odebrat. Tato hodnota tuto operaci reprezentuje a vlastně i uvádí, že se stala na recepci nějaká chyba.
- **celkem** - hodnota určující pouze rozdíl součtu vložených kreditů a bonusů s odebranými kredity. Určuje tedy celkový počet vložených kreditů a bonusů klientům ve zvoleném období.

Druhý sloupec reprezentuje počet odebraných kredtů v rámci jednotlivých rezervací. Jinými slovy, každá rezervace odebírá určitý počet kreditů, tyto kredity se sčítají a jejich znázornění je v tomto daném sloupci v prvním řádku. Druhý řádek, znázorňující počet vrácených kreditů, reprezentuje počet kreditů, které byly vráceny tím, že se daná rezervace smazala. Jinými slovy, pokud klient smaže rezervaci, jsou mu za ni vráceny zpět kredity, tyto kredity se sčítají a zobrazují na tomto příslušném řádku.

5.1.3 Statistika konkrétních rezervací a kreditů

Tato statistika reprezentuje počet rezervací jednotlivých přístrojů, cvičení, jejich kombinací a také počet objednaných produktů v obchodě. Jako názorný příklad můžeme pro tento typ využít obrázek (9). Zde je pro každý přístroj uvedeno

- **počet rezervací** - je to počet veškerých rezervací, včetně smazaných, na daný přístroj, cvičení či produkt v určitém zvoleném období
- **smazaných rezervací** - počet všech smazaných rezervací daného přístroje, cvičení či produktu.
- **aktuálních rezervací** - rozdíl mezi počtem rezervací a smazaných rezervací.

- **vytíženost přístroje** - hodnota uvádějící v procentech jak moc byl daný přístroj vytížen ve zvoleném období. Tato hodnota je vypočítána z podílu celkového možného počtu rezervací na daný přístroj a počtu konkrétních provedených rezervací. Poté ještě vynásobena 100 pro znázornění počtu procent.
- **odebraných kreditů** - hodnota spadající do druhého typu statistiky, a to do počtu kreditů, kde v tomto případě daná hodnata ukazuje, kolik bylo odebraných kreditů v rámci rezervace určitého přístroje, cvičení, či produktu.

5.2 Implementace

Než jsem začal statistiku implementovat, přemýšlel jsem jakým způsobem budu načítat data. Nabízely se mi dvě možnosti, a to

- vytvořit několik dotazů, které budou při každém načtení statistiky a výběru daného období prohledávat veškeré tabulky spojené s rezervacemi a kredity, spojovat je, vytvářet sumy apod., nebo
- vytvořit datový sklad, respektive další dvě tabulky, které se v průběhu vkládání rezervací či operací s kredity budou aktualizovat. K této akualizaci využiji materializovaného pohledu, který se mi nabízí opět ve dvou variantách, a to
  - občasná aktualizace tabulky pohledu (například 1 za týden) za pomocí SQL procedury
  - aktualizace ihned po každé rezervaci a operaci s kredity za pomocí triggerů

Jelikož tabulky pro rezervace a kreditové operace budou za pár let pro dané fitcentrum naplněné obrovským množstvím dat, připadalo mi jako nejlepší řešení zvolit druhou možnost pomocí datového skladu. Co se týče časového rozpětí aktualizace, nepřipadalo v úvahu aktualizovat občas materializovaný pohled, ale při každé provedené rezervaci. Statistika by měla být aktuální vždy, proto jsem využil triggery. Myslím si, že tato možnost přispěje i k menšímu vytížení serveru.

Pro vytvoření datového skladu jsem tedy vytvořil dvě nové tabulky, které budou „sbírat data“. Jedna s názvem **statistic_day_count**, která bude obsahovat data pro celkový počet rezervací a kreditů v jednotlivých dnech a druhá tabulka s názvem **statistic_pay_type** reprezentující data pro rezervace a kredity konkrétních přístrojů, cvičení či produktů v obchodě. Atributy těchto tabulek jsou opět v příloze v datovém slovníku.

Data v této tabulce jsou tedy rozdělena po jednotlivých dnech, každý řádek je jeden den a v každém tomto řádku jsou hodnoty reprezentující celkový počet rezervací a kreditů.

Druhá tabulka **statistic_pay_type** obsahuje ty samé atributy jako předchozí, kromě posledních dvou, protože neexistuje žádná vazba mezi přidanými a odebranými kredity zákazníkům a jednotlivými přístroji, cvičeními a produkty. V této tabulce ale naopak dva atributy přibyly.

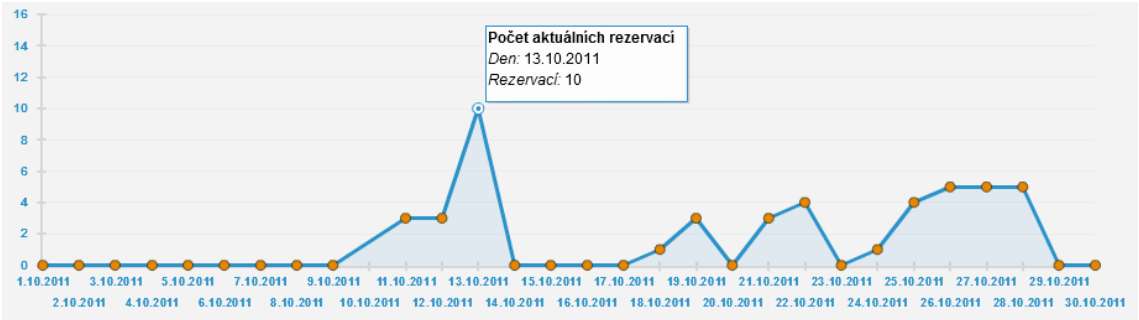
Tyto dvě tabulky se tedy plní při jakékoliv změně v rezervačním systému týkající se rezervací nebo kreditů. Například pokud si klient zarezervuje rezervaci na přístroj Power Plate, automaticky se ihned aktualizují obě dvě tabulky, a to tak, že se zvýší počet **resCount** o 1, zvýší se počet **creditCount** o počet kreditů, který je nastavený přístroji Power Plate. Pokud tato rezervace odebrala i něco z bonusů, přičte se i tato částka k **bonusCount**. Analogicky se toto dějě i při odebrání rezervace a při manipulaci s kredity. Je ale třeba při těchto operacích zamykat dané tabulky, protože kdyby najedou dva různí klienti provedli rezervace, celkové počty by ve výsledku nemusely odpovídat. Bylo tedy třeba toto ošetřit.

Jednotlivé plnění tabulek je prováděno za pomocí triggerů. Každá tabulka pro danou rezervaci či kredity tyto triggery obsahuje, a to konkrétně dva. Jeden pro vkládání a jeden pro editování dat. Neexistuje žádný trigger pro mazání, i když jsem o mazání rezervací již psal výše, a to z toho důvodu, že když se maže například rezervace, tak se nesmaže z databáze, ale je označena jako smazána (deleted), tudíž je třeba tabulku editovat.

Těmito triggery se tedy naplní dané tabulky a potom už se ve statistice jen vytvoří k zobrazení dat jednoduché SQL dotazy pro výběr dat. Samozřejmě, že pro výběr určitého období bylo třeba data sumarizovat přes jednotlivé dny, přesto je tato operace jednodušší, než kdyby dané tabulky neexistovaly.

5.2.1 Graf počtu rezervací

Pokud si některý z uživatelů v klientské nebo administrační části nechá vypsat statistiku za určité období, jsou mu tato data prezentována i ve formě grafu. Vzhledem k přehlednosti grafu a velkým rozdílům mezi hodnotami pro počet rezervací a kreditů, znázorňuji v grafu jen celkový počet rezervací. Uživatelé tak mohou ihned vidět, kolik rezervací měli v jednotlivých dnech, tak jak lze vidět na obrázku (10).



Obrázek 10: Graf počtu rezervací

Celý graf je vytvořený za pomocí nástroje FLEX, který je určený pro vývoj internetových aplikací připomínající klasické desktopové aplikace. Řadí se mezi RIA technologie. Celková implementace téměř nebyla náročná, protože ve FLEXu už byl určitým způsobem

nějaký graf přednastaven, nicméně jsem musel pozměnit spousty vlastností, aby vypadal graficky tak jako na obrázku (10). Bylo také třeba doprogramovat načítání externích dat, mimo to ale nic víc složitého.

6 Synchronizace s google kalendářem

Posledním modulem, který by mohl přijít vhod hlavně uživatelům využívajícím aplikace od společnosti Google, je synchronizace rezervací s Google kalendářem. Tedy, pokud určitý klient provede rezervaci buď na přístroje nebo cvičení, automaticky se mu v kalendáři vytvoří událost s nadpisem stejným, jako je název přístroje či cvičení v rezervovaný den a hodinu. Existují dva typy událostí v kalendáří, a to

- **veřejné** - jsou viditelné, kdokoliv se podívá na váš kalendář, tyto události jsou pouze a jen ke čtení a mají omezenější funkcionalitu,
- **soukromé** - jsou viditelné jen vámi a jejich funkcionalita je rozšířena.

Pro implementaci celé synchronizace jsem využil třídu, poskytující už zmiňovaný *Zend Framework*, který má v sobě určité funkce, pro přístup k Google API. Bylo tedy nutné tyto funkce pro přidání, načítání a mázání událostí v kalendáři prozkoumat a zjistit jejich funkčnost.

6.1 Možnosti přihlášení se k účtu google

Nejdůležitější částí synchronizace je přihlášní se k Google API kalendáři. Tato API umožňuje přístup k oběma dvěma typům událostí, jak k veřejným, tak k soukromým. Jediný rozdíl je v tom, že veřejné události nepotřebují autentizaci s google serverem. V tomto případě jsem vybral druhý typ události, protože nechci, aby různí lidé viděli, kdy a kde hodlám jít. Pokud bych tyto události chtěl někomu zveřejnit, tak pouze rodině, či některým známým, a to prostřednictvím nastavení google kalendáře přímo v účtu Google, kde si k sobě pozvu daného člověka.

Google kalendář tedy podporuje 3 typy přihlášení se na server, a to

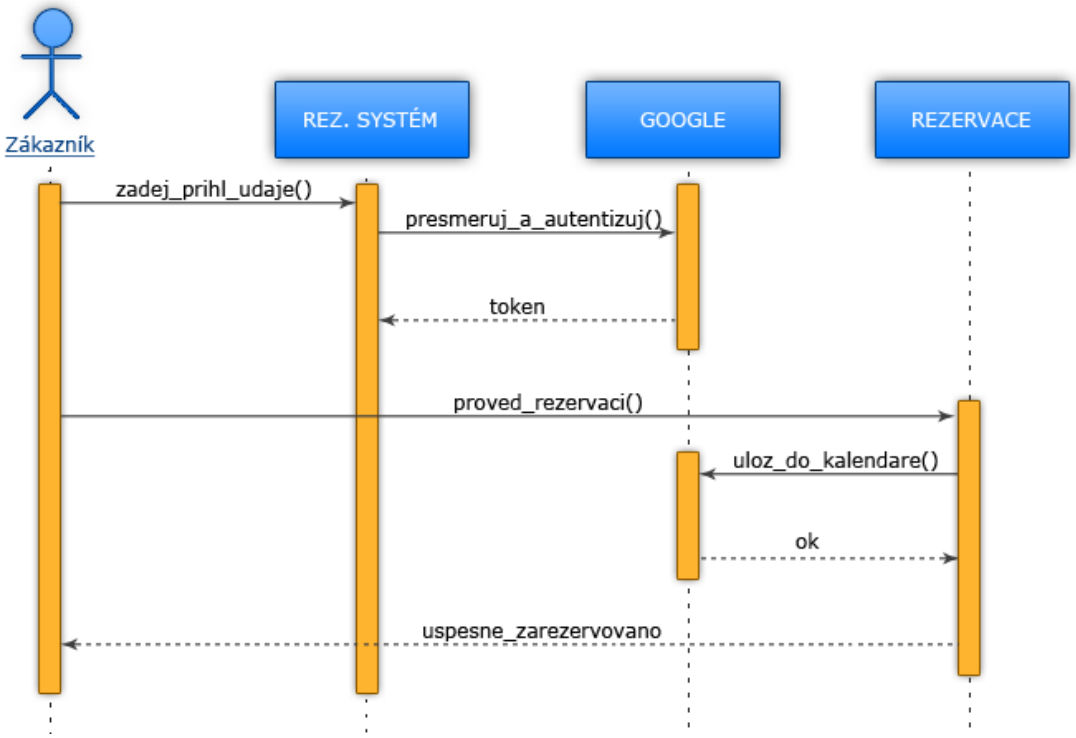
- **ClientAuth** - spojení se serverem Google je navázáno přímo za použití přihlašovacích údajů přes HTTP klienta. Na server Google jsou tedy poslány přihlašovací údaje, samozřejmě šifrovanou podobou, a také typ služby, kterou chceme využívat. Pro mě konkrétně kalendář. Na straně serveru proběhne autentizace a po úspěšném přihlášení se server pošle zpět povolení daný kalendář využívat, včetně všech jeho funkcí. K využívání tohoto příhlášení je tedy nuté mít ve své aplikaci uloženy veškeré přihlašovací údaje všech klientů ke svým účtům Google, aby mohli své rezervace ukládat do Google kalendáře.
- **AuthSub** - toto přihlášení je narozdíl od předchozího rozdílné v tom, že se na server neposílají žádně přihlašovací údaje a nemusíte u sebe mít uložené veškeré přístupové údaje klientů k jejich Google účtu. Pokud provedete rezervaci, tak jste ze systému přesměrování přímo na server Google, kde se musíte přihlásit, je Vám přidělen určitý *token* a potom jste přesměrování zpět do rezervačního systému, kde

se daný přidělený *token* využije k přístupu do Google API kalendáře. Navíc se tento *token* uloží do *session* a když budete provádět rezervace a *session* bude stále naplněná tokenem, nebudete opět přesměrování na Google a nebudete se muset opět přihlásit, ale budete jeho služeb nadále využívat. Jakmile *session* vyprší, je třeba se znovu přihlásit.

- **MagicCookie** - přihlášení, které je z daných tří nejjednodušší co se implementace týče. Nicméně díky němu máte přístup k událostem kalendáře jen pro čtení, nemůžete s nimi nijak manipulovat.

6.2 Konkrétní využití synchronizace

V mém případě jsem pro autentizaci s Google serverem využil druhou možnost, a to z toho důvodu, že nechci uchovávat ve svém systému veškeré přihlašovací údaje všech registrovaných klientů. Bylo by to vůči nim neseriózní, a proto je lepší, když budou chtít tuto službu využít, aby je to pro přihlášení k google nejdříve odkázalo na server google a potom zpět do mého systému.



Obrázek 11: Sekvenční diagram synchronizace s google kalendářem

Jestliže chtějí klienti ve své klientské části rezervačního systému využívat synchronizaci s google kalendářem, ihned při přihlášení se do systému mají možnost zaškrtnou

políčko s názvem „Synchronizovat rezervace s google kalendářem“. Pokud jej zašrtnou, jsou přesměrování na google, tam se přihlásí, potvrdí, že umožňují přístup k účtu na google a poté jsou přesměrování zpět na systém (včetně tokenu), zde se už ale nemusí opět přihlašovat údaji pro rezervační systém, ale ihned už se objeví v rozvrhu pro rezervace na přístroje. Nyní při každé rezervaci systém využívá uložený *token* v session k přihlášení se do API google a rezervace se ukládají do kalendáře. Pokud klient rezervaci smaže, smaže se i v kalendáři. Na obrázku (11) můžeme lépe vidět jak probíhá vkládání rezervací do kalendáře. Mazání probíhá analogicky.

Nejprve jsem si myslel, že budu testovat, jestli v období, kdy se klient rezervuje, už existuje nějaká událost v google kalendáři. Po lepším zvážení daného problému jsem si uvědomil, že je možné, že klient bude mít například na 10:30 upomínku, že má někomu zavolat a tím pádem by se klidně i na 10:30 mohl rezervovat a těsně před cvičením danému člověku zavolat. Proto jsem tuto kontrolu nevyužil a je tedy možné v jeden den a hodinu mít více událostí.

Bohužel ale neexistuje možnost, že by si klient zarezervoval rezervaci přes rezervační systém, poté ji smazal v google kalendáři a toto smazání by se provedlo i v systému. Taková zpětná vazba v API google neexistuje. Je třeba si danou rezervaci dodatečně smazat ještě v systému.

7 State-of-the-art přístupy, které se používají pro autentizaci a autorizaci

Autentizace je proces ověření identity subjektu. Autentizace patří k bezpečnostním opatřením a zajišťuje ochranu před falšováním identity, kdy se subjekt vydává za někoho, kým není. Většinou se používá znalost jména a hesla, v lepším případě se používá certifikát či biometrie (otisky prstů apod.). Po dokončení autentizace obvykle následuje autorizace, což je souhlas, schválení, umožnění přístupu či provedení konkrétní operace daným subjektem.

7.1 Basic authentication

Mezi nejzákadnější a zatím nejpoužívanější autentizace patří tzv. *Basic authentication* prostřednictvím protokolu HTTP, kde pro přístup ke službě využíváte své uživatelské jméno a heslo. Tyto údaje se, samozřejmě v zakódované podobě, uloží na příslušnou dobu do *SESSION*[6] v klientské části. Uživatel je tedy stále přihlášen, dokud *SESSION* nevyprší. Tuto autentizaci používá spousta e-shopů a dalších běžných webů. Pro tento případ je ale nutná registrace na stránkách, kde vám je dané uživatelské jméno a heslo přiděleno, případně si ho potom můžete změnit v samotné službě.

7.1.1 Výhody protokolu

Výhodou této autentizace je, že si daná služba eviduje vlastní seznam uživatelů a správa tohoto seznamu může být navržena do posledního detailu k potřebám dané služby. Služba má tedy nad veškerými uživateli maximální kontrolu.

7.1.2 Nevýhody protokolu

Nevýhodou, co se uživatelů týče, je to, že při registraci do různých služeb si musí uživatelé pamatovat neskutečné množství přihlašovacích údaju, což je někdy nad lidské síly. Pravdou je, že se objevují různé pomůcky, které si za vás pamatují přihlašovací údaje v prohlížeči. Pokud ale přistupujete do služby i z jiných počítačů, tak už tyto údaje uložené nemáte a musíte trápit svůj mozek, aby jste si na ně opět vzpomněli.

Další nevýhodou je nepřenositelnost dané identity a její nejednoznačnost. Jedná se zejména o to, když přidáváte nějaké příspěvky do fóra, případně do blogů a jiných částí webů, neboť pokaždé můžete mít jiné uživatelské jméno. A tak se může stát, že když je na jednom serveru registrovaný uživatel *petr* a na druhém si já založím uživatele *petr* a oba servery obsahují podobnou tématiku, potom je možné, že uživatel *petr* na prvním serveru bude pomlouvat *petra* z druhého serveru a tím pádem ho určitým působem poškozovat.

[6] prostředek (jmenný prostor) pro uchování dat (uživatelské jméno, heslo) v prostředí webu a přechodu z jedné části webu na druhý

Kvůli těmto nevýhodám se začalo přemýšlet, jak vše sjednotit do jedinného účtu a vznikaly protokoly s označením *single sign-on*. Mezi ně patří například protokol *Kerberos*, *OpenID* a *LiveID*. Těchto protokolů je samozřejmě více [1], zde však probereme pouze tyto tři.

Na konci této kapitoly budu ještě rozebírat jeden protokol, a to *OAuth*, který už nepatří mezi *single sign-on* protokoly, ale také využívá jednoté přihlášení a hlavně umožňuje otevřeně přistupovat k API dané internetové služby, a proto ho v dnešní době začaly využívat společenosti jako Google, Facebook, Twitter apod.

7.2 Kerberos

Kerberos je viz [2] síťový autentizační protokol, který byl vytvořen primárně pro model klient-server a poskytuje vzájemnou autentizaci. Klient i server si ověří identitu své protistrany. Umožňuje komunikujícímu v nezabezpečené síti prokázat bezpečně svoji identitu někomu dalšímu. Původní implementaci Kerbera najdeme u MIT [2]. Další implementaci Kerbera najdeme u Microsoftu [3], kde se od MS Windows 2000 používá jako autentizační mechanizmus v Active Directory.

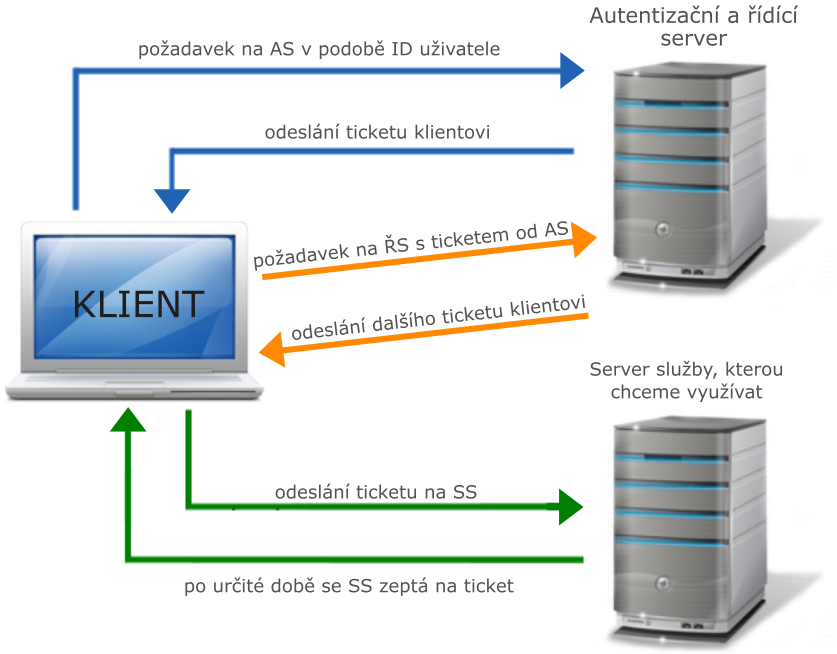
7.2.1 Princip funkčnosti

Pro autentizaci využívá prostředky třetí strany, a to konkrétně autentizační server (dále jako AS) a řídící server (dále jako ŘS), které poskytují tzv. tiket, ten se zasílá zpět klientovi jako ověření jeho identity. Po ověření má klient přístup k serveru služby (dále jako SS). Popišme si zjednodušeně jak to tedy funguje.

- Klient požaduje přístup k SS, a proto vyplní užvatelské jméno a heslo k přihlášení se do služby.
- U klienta se dané vložené heslo uloží a na AS se zašle jméno klienta. AS zjistí, jestli daného klienta má uloženého ve své databázi a pokud ano, z databáze vybere jeho heslo, vytvoří pomocí tohoto hesla hash a ten zašle jako tiket zpět klientovi.
- Klient obdrží ticket s heslem, dešifruje ho a pokud se heslo uložené u klienta rovná heslu zaslaného AS, odešle se další ticket od klienta do ŘS s tajným klíčem, který znají pouze AS a ŘS.
- Tím se zajistí, že ŘS pozná daného uživatele, dochází tedy k autorizaci a pošle klientovi oprávnění používat SS prostřednictvím dalšího tajného klíče.
- Dále se provádí také ověření, jestli SS je opravdu ta služba, kterou daný uživatel vyžaduje. SS a ŘS znají svůj tajný klíč zaslaný klientovi v minulém kroku.
- Pokud SS přijme klíč, klient má přístup do služby.

- Tento přístup je ale časově omezen a během tohoto času není třeba se znovu autentizovat. Jakmile ale časové období vyprší, stačí poté znovu použít už obdržený tiket a pomocí něj se znovu autentizovat.

Celkový průběh autentizace a autorizace pomocí protokolu Kerberos je na obrázku (12).



Obrázek 12: Ukázka funkčnosti protokolu Kerberos

7.2.2 Výhody protokolu Kerberos

- Jednou z výhod tohoto protokolu je, že pro autentizaci se heslo zadané klientem v přihlašovacím formuláři nikam neposílá, posílá se pouze jméno. Autentizační server na základě tohoto jména zašle v zakódované podobě příslušné heslo ze své databáze klientovi, u kterého se dané heslo porovná s heslem vloženým do formuláře, čímž se zjistí, jestli je to opravdu on.
- Další výhodou je, že autentizace je prováděna i opačným směrem, provádí se test, jestli daná služba, na kterou se chce klient připojit, je ta, kterou vyžaduje.

7.2.3 Nevýhody protokolu Kerberos

Nevýhodou protokolu Kerberos je, že je potřeba neustáleho běhu AS a ŘS k autentikaci uživatele. Pokud tyto servery nejedou, nejde se ani do služby přihlásit.

Dále Kerberos přísně kontroluje časovou synchronizaci mezi klientem a AS a ŘS. Pokud tato synchronizace není kvalitní, autentizace selže.

Navíc pokud by z nějakého důvodu byla narušena autentizační struktura přes AS a ŘS, mohl by se daný uživatel vydávat za kohokoliv by chtěl.

7.3 OpenID

OpenID je, jak uvádí wikipedie, otevřený standard popisující decentralizovaný způsob autentizace uživatelů, který odstraňuje potřebu na straně provozovatele služby poskytovat a vyvíjet vlastní systémy pro autentizaci, a který rovněž samotným uživatelům služby umožňuje konsolidaci jejich digitálních identit. Jednoduše řečeno, vyberete si konkrétního poskytovatele OpenID (například Google), u kterého máte vytvořený účet. Máte u něj uloženo své uživatelské jméno a heslo a díky těmto údajům dostanete své OpenID a využíváte jej všude, kde je jeho podpora při autentizaci klientů. Stačí se jen vždy u daného poskytovatele pomocí jména a hesla přihlásit.

OpenID má tvar běžného URL, je to pouze vygenerovaný dlouhý řetězec. Pokud již máte účet například u již zmiňovaného google, případně u českého vyhledávače seznam, tak své OpenID už dávno máte a ani o tom nevíte. Například, jestliže máte účet od google, máte vlastně účet i od google+ a pokud se budete nacházet na své profilové stránce sociální sítě google+, tak OpenID google je právě URL adresa vaší profilové stránky, konkrétně moje je *https://plus.google.com/118016616191132303907*.

U seznamu je například vaše OpenID řetezec vaše emailová adresa, pouze se změní symbol zavináče na *.id.*, takže například meca.petr.id.seznam.cz.

Pokud byste nechtěli využívat OpenID od žádného z těchto poskytovatelů, nabízí se možnost využít dalších, a to například *MyOpenID* [4] nebo *MojeID* [5]. Případně se sami můžete stát poskytovateli dané služby a nabízet ji ostatním. Je třeba si jen pořádně přečíst dokumentaci k OpenID a provést správnou implementaci.

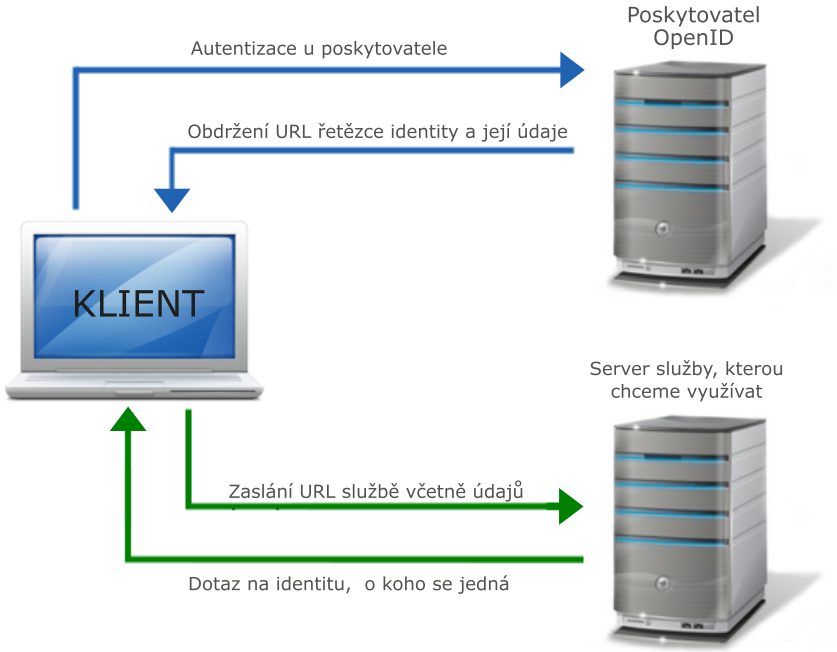
7.3.1 Funkčnost OpenID

- Na stránkách, kde se chcete přihlásit, vyplníte ve formuláři kolonku pro OpenID řetezec.
- Systém vás přesměruje na server poskytovatele služby OpenID.
- Budete vyzvání k zadání hesla. V budoucnu je možné tento krok přeskočit tím, že potvrdíte, aby všechny požadavky z vašich stránek vztahující se k vašemu OpenID

systém bral automaticky za schválené. Ale z hlediska větší bezpečnosti doporučuji pokaždé nechat možnost zadávat heslo.

- Pokud bude heslo ověřeno, zobrazí se tlačítko, které vás přesměruje zpět na vaše stránky s informací, že všechno proběhlo v pořádku.
- Systém tento požadavek zpracuje. Pokud to je vaše první přihlášení, tak vás zároveň zaregistruje. Pokud jste se někdy v minulosti tímto způsobem přihlašovali, tak vás rovnou přihlásí.

Funkčnost OpenID můžeme opět pochopit ve zjednodušené formě z obrázku (13).



Obrázek 13: Ukázka funkčnosti protokolu OpenID

Google OpenID jsem v praxi vyzkoušel na již zmiňovaném serveru seznam.cz. Stačí najet na adresu login.seznam.cz, vpravo vybrat přihlášení pomocí OpenID a vložit dané URL. Seznam vás nejdříve upozorní na to, že si chce toto nové OpenID přiřadit k vašemu účtu, takže se musíte naposledy přihlásit na seznam pod seznamovským účtem a vaše OpenID je přidáno do seznamu. Nyní pokud se pokusíte znovu se přihlásit, automaticky se dostanete do vašeho účtu seznamu. Na ostatních serverech může být funkčnost trochu jiná, a to v tom ohledu, že vás nebudou žádat o přiřazení OpenID k vašemu účtu na daném serveru, registrace se provede automaticky.

7.3.2 Výhody OpenID

- Protokol OpenID je od začátku koncipován jako otevřený – tedy za jeho použití nemusíte platit, jeho specifikace je plně k dispozici a není zatížená licenčně, ani nepatří nějaké firmě.
- Poskytovatelů OpenID může být spousta a je jen na vás, kterého si vyberete. Můžete se také stát sami poskytovateli.

7.3.3 Nevýhody OpenID

Pokud by někdo vytvořil falešnou stránku poskytovatele, mohli bychom mu omylem zadat své jméno a heslo, které tím, že je globální, má velkou hodnotu. Pokud se stránkou poskytovatele komunikujeme pomocí protokolu https (a neignorujeme bezpečnostní výstrahy prohlížece při neplatnosti certifikátu), pak je možnost phishingu úplně eliminována.

7.4 LiveID

LiveID je protokol vyvíjen firmou Microsoft jako jednotné přihlášení se v rámci veškerých internetových služeb, které Microsoft nabízí. Také je ale možné tuto službu využít na vlastním webu.

Tento protokol už z názvů nabádá k tomu, že bude mít stejnou funkčnost jako OpenID, ale není tomu tak. Základ mají stejný, a to, že reprezentují jednotné přihlášení se do příslušné služby na internetu (nemusí to být jen v rámci windows aplikací), nicméně rozdíl oproti OpenID je v tom, že

- jednoznačným identifikátorem je u LiveID emailová adresa, kokrétně něco@hotmail.com nebo něco@windows.live.com, je ale možné si vytvořit účet i s vlastní emailovou adresou.
- Z programátorského hlediska má Live ID další odlišnost, a to potřebu pro každou službu vygenerovat na stránkách Microsoftu identifikátor aplikace, takzvaný AppID.
- Dále Live ID není omezeno pouze na webové aplikace, jeho pomocí se mohou uživatelé přihlašovat i k desktopovým aplikacím. Microsoft dodává i .NET knihovny pro vývojáře desktopových aplikací, které implementují Live ID přihlašování i pro C#.
- Posledním a tím nejdůležitějším rozdílem je, že LiveID není decentralizovaný, což znamená, že přihlašovací účet pro získání jednotné identity získáte pouze u Microsoftu, nikde jinde.

Nicméně Microsoft už má v dnešní době podporu OpenID, která je postavena na platformě LiveID [6].

7.5 OAuth

OAuth (Open Authentication) [7] je otevřený protokol, jehož cílem je poskytnout bezpečnou autentizaci a autorizaci oproti API různých služeb. Jeho využití je rozsáhlé, lze jej využít jak pro desktopové, mobilní, tak i webové aplikace. Provozovatelům služby, která OAuth identity poskytuje, dává OAuth možnost sdílet uživatelská data a identity, aniž by uživatelé museli prozrazovat své heslo komukoliv dalšímu. Můžete sdílet své fotky, videa, články, prostě veškeré své údaje.

Různé webové služby, zejména sociální sítě, nabízejí API pro přístup ke svým funkcím, vývojáři tedy mohou tvořit aplikace, které tyto funkce využívají. Lze tak vytvářet aplikace, které mohou publikovat přes Twitter nebo Facebook, přistupovat k fotografiím v albu apod. Je třeba ale tento přístup ověřovat uživatelským souhlasem a přihlášením se do této sociální sítě.

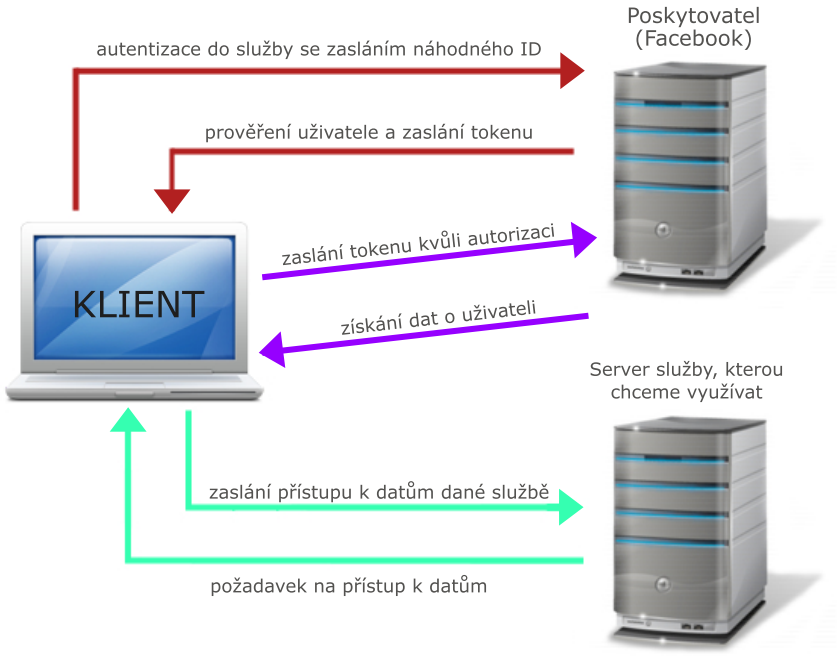
7.5.1 Funkčnost OAuth

Tuto službu jsem ve svém rezervačním systému také využil, konkrétně prostřednictvím sociální sítě Facebook, kdy se klienti určitého fitcentra mohou do systému přihlásit pomocí přihlašovacího formuláře umístěného na už zmíněné sociální síti. v systému lze poté zobrazit například fotku daného uživatele, či mu pomocí jeho data narození zaslat email s přáním apod. Níže si zjednodušeně popišme jak daná služba funguje.

- Nejdříve je tedy třeba si založit novou aplikaci na facebooku, konkrétně zde `https://developers.facebook.com/apps`.
- Po jejím založení obdržíte ID dané aplikace a její tajný klíč. Navíc musíte ještě vyplnit url adresu webu, kde budete chtít autentizaci provádět.
- Jakmile budete mít hotovou implementaci autentizace s facebookem na vašem webu, bude-li se klient chtít přihlásit do vaší aplikace, bude přesměrován nejdříve na přihlašovací formulář Facebooku. Před tím se ale ještě u klienta uloží do SESSION náhodně vygenerované ID, které se také při přihlášení zasílá do Facebooku.
- Uživatel se přihlásí a pokud ve vaší aplikaci budete chtít využívat některé jeho informace, jako například klientův email, fotky, videa apod., musí vám tyto informace povolit v následujícím formuláři.
- Poté bude přesměrován do vaší aplikace s jeho vygenerovaným ID a zkontroluje se, jestli se toto ID rovná tomu, které on zaslal na začátku. Navíc Facebook ještě zasílá tzv. *token*.

- Tento *token* se zašle opět Facebooku společně ještě s tajným klíčem, který vám byl přidělen při založení aplikace, jako autorizace, že se opravdu jedná o vás a Facebook vám následně pošle veškeré informace o uživateli, vy je pak můžete začít ve své aplikaci využívat.

Celkovou funkčnost protokolu si opět můžeme znázornit na obrázku (14).



Obrázek 14: Ukázka funkčnosti protokolu OAuth

7.5.2 Výhody OAuth

- Největší výhodou tohoto protokolu je, že je otevřený, a že díky němu může určitá internetová služba nabízet své API, tuto službu poté mohou využívat třetí strany pro své menší aplikace. V dnešní době tuto službu hojně využívají společnosti jako Facebook, Google, Twitter apod.
- Protokol lze využívat jak pro internetové aplikace, tak i pro desktopové a mobilní telefony.

7.6 Rozdíl mezi single sign-on a OAuth

Rozdíl tedy mezi protokoly *single sign-on* a *OAuth* je ten, že *OAuth* poskytuje otevřený přístup k celkové API dané služby, aniž byste sdělovali přístupové údaje třetí straně, ve které chcete danou API využívat.

Single sign-on protokoly pouze určují identitu klienta, zaručují, že přihlášený uživatel jste opravdu vy a nikdo jiný. Také o vás zasílají určité informace, ale neposkytují tak otevřený přístup jako *OAuth*.

Dovolím si zde ještě vložit tabulku rozdílu mezi těmito protokoly, kterou jsem si vypůjčil ze serveru root.cz[7], konkrétně mezi protokoly OpenID a OAuth.

Rys	**OpenID**	**OAuth**
Rozsah	Definuje komunikační protokol, výměnu informací, rozsah informací, obsahuje i několik úrovní autentizace	Definuje jen komunikační protokol pro autentizaci; veškeré další zjišťování údajů musí proběhnout přes proprietární API té které služby
Architektura	Distribuovaný autentizační systém, kde jsou uživatelské účty nezávislé na službách, které je využívají	Zajištění autentizace uživatelů konkrétní služby
Problém, který řeší	Obecné ověření uživatele; poskytnutí některých informací o něm; zabezpečení přihlašování	Ověření uživatele kvůli přístupu k důvěrným funkcím API webových služeb.
Topologie	„Uživatelsko-centrická“ (důraz je kladen na identitu uživatele)	„Službo-centrická“ (důraz je kladen na samotný fakt ověření)
Uživatelský účet	Může být u libovolného poskytovatele; lze jej použít v různých službách	Uživatel musí být nejprve zaregistrován u dané služby

Tabulka 1: Rozdíly mezi protokolem OpenID a OAuth

Za zmínku ještě stojí uvést, že Google vyvíjí „Hybridní“ protokol, který obě dvě tyto technologie kombinuje dohromady [8].

[7] http://zdrojak.root.cz/clanky/openid-vs-oauth-bitva-ktera-se-nekona/

8 Rozšíření pro SQL Server

Jak už bylo napsáno v úvodu, rezervační systém využíva databázi MySQL. V této kapitole se budu zaobírat rozšířením této databáze pro SQL Server, a to konkrétně o rozšíření datové vrstvy systému, testovací vytížení a odladění fyzického návrhu databáze a nakonec audit databázových operací. SQL Server obsahuje spoustu nástrojů, které využijeme k vytvoření a otestování všech těchto částí. Samozřejmě bychom si mohli vytvořit například audit i v MySQL, bylo by ale třeba vytořit spoustu dalších tabulek, do kterých bychom ukládali veškeré operace administrátorů a také by bylo třeba veškeré operace otestovat, jestli fungují správně. Nicméně SQL Server už toto vše v sobě obsahuje, a proto nám v tomto případě ušetří spoustu času.

8.1 Rozšíření datové vrstvy systému

Jako první věc, kterou musíme provést pro využívání SQL Serveru, je přesun kompletní MySQL databáze na daný server a s tím spojené SQL dotazy prováděné při využívání systému. S tím souvisí spousta problémů, a to v tom, že SQL Server má v určitých dotazech na databázi jinou syntaxi než MySQL nebo například některé příkazy vůbec neobsahuje.

8.1.1 Import tabulek

V první řadě je třeba exportovat veškeré tabulky z MySQL databáze a tady hned nacházíme rozdílnou syntaxi SQL dotazu. Po exportu jedné tabulky z MySql dostaneme následující dotaz

```
CREATE TABLE reservation (
  reservationID int(11) NOT NULL AUTO_INCREMENT,
  customerID int(11) NOT NULL,
  date datetime NOT NULL,
  equipmentID int(11) NOT NULL,
  price int(11) DEFAULT '0',
  note text NOT NULL,
  trainer tinyint (1) NOT NULL DEFAULT '0',
  PRIMARY KEY (reservationID),
  KEY customerID (customerID),
  KEY date (date),
  KEY equipmentID (equipmentID),
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=185 ;
```

Výpis 1: SQL dotaz pro vytvoření tabulky v MySQL

Pokud bychom ho chtěli spustit v SQL Serveru, narazíme na několik chyb, a to

- u *int(11)* nesmí být určení délky intigeru, je třeba tedy odstranit celou závorku u všech *INT* a *TINYINT*,
- v SQL Serveru neexistuje *AUTO_INCREMENT* , je třeba ho nahradit příkazem *IDENTITY(1,1)*, kde čísla v závorce označují počáteční hodnotu a o kolik se daná hodnota má zvýšit při každém vložení nového záznamu,
- příkaz *KEY* také neexistuje, je třeba ho úplně odstranit a indexy vložit pomocí dalších dotazů,
- nakonec poslední řádek, který určuje typ tabulky, kódování je třeba také odstranit.

Výsledný SQL dotaz by tedy pro SQL Server vypadal takto

```
CREATE TABLE reservation (
  reservationID int NOT NULL IDENTITY(1,1),
  customerID int NOT NULL,
  date datetime NOT NULL,
  equipmentID int NOT NULL,
  price int DEFAULT '0',
  note text NOT NULL,
  trainer tinyint NOT NULL DEFAULT '0',
  PRIMARY KEY (reservationID)
);

CREATE INDEX customerID ON reservation (customerID);
CREATE INDEX equipmentID ON reservation (equipmentID);
CREATE INDEX date ON reservation (date);
```

Výpis 2: SQL dotaz pro vytvoření tabulky v SQL Serveru

Tímto předěláním SQL dotazu můžeme vložit veškeré tabulky systému do SQL Serveru a co se databáze týče, máme ji připravenou pro využití Fitsystému. Nyní je třeba nepatrně přepracovat některé dotazy pro funkčnost systému a ukládání dat do databáze.

8.1.2 Využívání data

Ve spoustě dotazech a hlavně v triggerech byl problém s formátem data. Jednalo se o to, že v triggeru SQL Server datum vkládal ve formátu

Jan 29 2012 7:30AM

což *DATETIME* formát v databázi nepodporuje. Bylo třeba toto datum překonvertovat, a to následujícím dotazem

REPLACE(CONVERT(VARCHAR(10), @date, 111), '/', '-');

kde *@date* je vkládané datum, funkce *CONVERT* překonvertuje dané datum podle kódu *111* do formátu *YYYY/MM/DD*, neexistuje totiž kód, který by přímo překonvertoval datum do cíleného formátu *YYYY-MM-DD*. První formát tedy přes funkci *REPLACE* přepíšeme na už výsledný *YYYY-MM-DD*, který vyhovuje naší databázi.

8.1.3 příkaz LIMIT

Velkým překvapením bylo, že SQL Server neobsahuje v dotazech příkaz LIMIT, který v MySQL databázi slouží k vypsání jen určitého počtu řádků tabulky. Navíc si můžete určit, od kterého řádku chcete začít vybírat data a kolik. Jeho umístění v SQL dotazech se provádí až na konec dotazu, konkrétně

SELECT m,n FROM tabulka ORDER BY at LIMIT f,t. (1)

Příkaz *LIMIT* můžeme využívat ve dvou variantách, a to

LIMIT f (2)

případně

LIMIT f,t (3)

kde v prvním případě

- *f* je počet vybraných prvních *f* řádku tabulky

a v druhém případě

- *f* je f-tý řádek v tabulce od kterého se začnou vybírat data a
- *t* je počet řádků, které se mají vybrat.

V SQL Serveru, jak už je psáno výše, tento *LIMIT* příkaz neexistuje. Je nutno jej nahradit jiným příkazem, a to pro obě varianty *LIMITu* pokaždé jiným, respektive celým SQL dotazem. První variantu ve výrazu (2) nahradíme příkazem

TOP f

který se naopak od *LIMITu* nachází přímo za příkazem *SELECT*, kde hodnota *f* znamená počet vybraných prvních *f* řádku tabulky. Druhou variantu zobrazenou ve výrazu (3) musíme ale rozšířit o příslušný SQL dotaz. Využijeme tedy už zmíněný dotaz (1), který pro funkčnost v SQL Serveru přepíšeme na

```
SELECT m,n FROM (
    SELECT m,n,ROW_NUMBER() OVER (ORDER BY at DESC ) as row FROM table
)  a WHERE row > f and row <= t.
```

Výpis 3: SQL dotaz nahrazující LIMIT z MySQL

Jak lze vidět z předchozího dotazu bylo třeba, aby se (1) stal vnořeným dotazem a navíc k příkazu *SELECT* bylo třeba dodat výběr řádků dané tabulky, konkrétně

$$ROW\_NUMBER()\ OVER\ (ORDER\ BY\ at\ DESC\ )\ as\ row\ ,$$

který zároveň obsahuje i příkaz *ORDER BY*, jenž byl v dotazu (1) až na konci. Zde se nachází přímo v příkazu *SELECT*. Výraz

$$a\ WHERE\ row > f\ and\ row <= t$$

vyjadřuje potom právě téměř to samé jako (3), kde

- *f* je f-tý řádek v tabulce, od kterého se začnou vybírat data a
- *t* je t-tý řádek pro který se dané řádky vypíšou. Zde je malá změna oproti (3)

8.1.4 příkaz GROUP BY

U tohoto příkazu se naskytly problémy s tím, že pokud pomocí příkazu *SELECT* vybereme nějaké atributy z databáze, je třeba i tyto atributy zařadit k příkazu *GROUP BY*. Uveďme si příklady využití *GROUP BY* v MySQL a SQL Serveru. V MySQL by dotaz vypadal následovně

$$SELECT\ COUNT(a),b,c\ FROM\ table\ GROUP\ BY\ a.$$

V SQL Serveru musíme do *GROUP BY* zařadit i hodnoty *b* a *c*, tudíž

$$SELECT\ COUNT(a),b,c\ FROM\ table\ GROUP\ BY\ a,b,c,$$

jinak dotaz nebude fungovat. Objevuje se zde ale ještě jeden problém, a to v případě, pokud by hodnota *b* nebo *c* byla typu *TEXT*, jelikož *GROUP BY* nelze na *TEXT* využít, SQL Server toto bohužel nepodporuje. Proto je třeba ještě hodnotu *b* nebo *c* překonvertovat na typ *VARCHAR* o maximální šířce následujícím příkazem

CAST(b AS VARCHAR(8000)) as b.

8.1.5 Triggery

Nejnáročnější předělání SQL dotazů bylo v rámci triggerů, které se využívají pro vyhodnocování statistiky veškerých rezervací a kreditových operací. Zde se nacházela spousta věcí, které potřebovaly změnu.

Nejjednodušší opravou v rámci triggeru bylo definovaní proměnných. V MySql se nemusí před proměnnou psát @ jak tomu bývá v SQL Serveru, proto bylo třeba tento znak ke každé proměnné doplnit.

DECLARE @variable INT

Dalším zjištěním bylo, že SQL Server nepodporuje *BEFORE* trigger, pouze *AFTER*, takže bylo třeba nějak vyřešit načítání dat do triggeru například před vložením rezervace do sytému. Po dlouhém bádání jsem na stránkách SQL Serveru zjistil, že existují tabulky *inserted* a *deleted*, které nám daný *BEFORE* trigger nahradí. Tyto tabulky slouží pro uchování dat, které se právě do konkrétní tabulky v databázi vkládají, případně editují nebo mažou. V MySQL jsme nově příchozí hodnoty do triggeru označovali jako

NEW.variable.

V SQL Serveru už *NEW* použít nemůžeme a využijeme k tomu tabulku inserted, konkrétně

SELECT @variable = ins.variable FROM inserted ins. (4)

Vyskytl se zde ale další problém, a to ten, že dotazy (4) vybíraly z databáze i celkové počty řádků dané tabulky a tato hodnota bránila ve výpisu potřebných dat. Proto bylo nutné ještě před dotazy (4) dát příkaz

SET NOCOUNT ON,

který zakázal vybírání počtu řádků.

8.2 Testovací vytížení a odladění fyzického návrhu databáze

Nyní se přesuňme do druhé fáze rozšíření pro SQL Server, a to konkrétně do testování vytíženosti databáze. Budeme k tomu využívat nástroje SQL Serveru jako je *SQL Profiler* a *Execution plan*, ale také volně dostupné nástroje RML[8]. Úkolem bude najít takové SQL dotazy, které zpomalují chod systému, jejich provedení trvá dlouho a mají velké množství logických operací, jako *READ* a *WRITE*. Dále bych chtěl zdůraznit, že celkové vytížení se zkoumá pouze v klientské části sytému, jen jeden dotaz z administrační části, a to proto, že v klientské části se provádi nejvíce operací. Může zde být najednou připojených 20 lidí, kteří mohou najednou provádět rezervace na rozdí od administrační části, kde je připojen jen jeden administrátor a převážně má po celou dobu otevřený dnešní rozvrh hodin. Z administrační části jsem tedy vybral jen jeden dotaz, a to na zobrazení rozvrhu, ostatní dotazy byly opomenuty.

8.2.1 Návrh virtuálních cest pro průchod uživatelů systémem

K tomu abychom mohli řádně otestovat rezervační systém, je třeba si ze začátku navrhnou virtuální průchod daného systému, což znamená naprogramovat klasické chování klientů v systému a provádět přesně ty samé SQL dotazy nad databází jako by je prováděli oni. Pro lepší orientaci a hlavně pro rychlejší naprogramování tohoto průchodu bylo dobré vytvořit určitý graf, který by daný problém řešil.

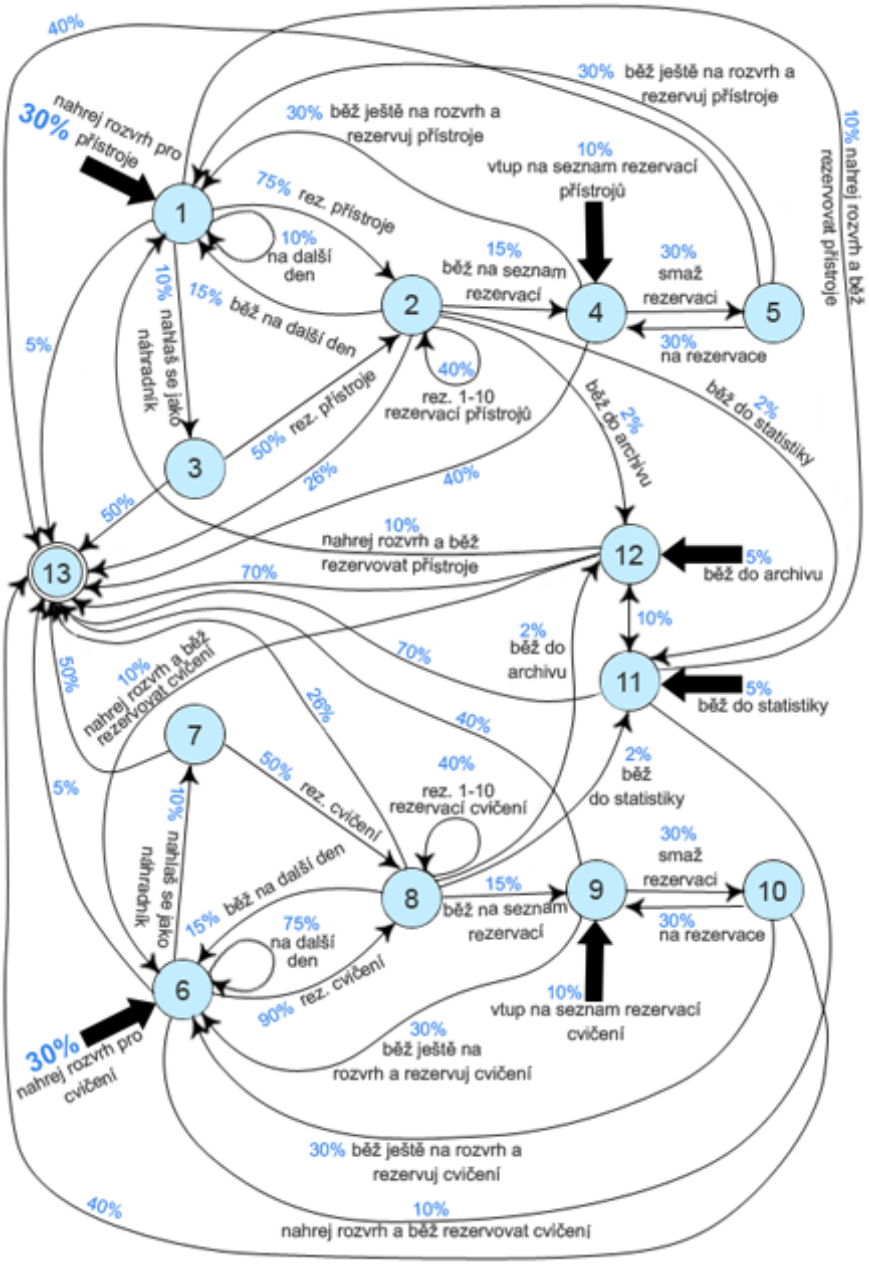
Na obrázku (15) můžeme tedy vidět daný průchod systému, kde ohodnocením hrany je procentuální počet virtuálních klientů procházjících přes tuto hranu s 6 vstupnípmi uzly a jedním koncovým uzlem, do kterého se můžeme dostat ze všech ostatních uzlů. Procentuální ohodnocení není nijak odvozeno ze zkoumání chování lidí ve fitcentru, je to pouze můj logický nahled na danou problematiku a myslím si, že takto by se nejpravděpodobněji mohl chovat člověk při spravování svých rezervací.

Vstupními uzly tedy jsou

- **rezervace na přístroje**, kde procentuální pravděpodobnost, že daný klient začne v této části, je nastavena na **35%**,
- **rezervace na cvičení**, zde je pravděpodobnost nastavena na stejnou hodnotu **35%**, a to z toho důvodu, že tyto rezervace jsou podobné rezervacím na přístroje, a proto na ně může přijít stejný počet lidi,
- **seznam rezervací přístrojů**. Zde už je procentuální počet menší, protože lidé se zejména chodí rezervovat, méně už se dívat, kdy si vlastně dané rezervace zarezervovali, případně si je přijdou smazat, proto počet procent je zadán na hodnotu **10%**
- **seznam rezervací cvičení**, opět je zde stejná hodnota jako u seznamu rezervací na přístroje, tedy **10%**,

[8] http://support.microsoft.com/kb/944837/en-us?fr=1

- **statistika** a **archiv operací**, do kterých přijde už jen zbylých **10%**, do každého tedy **5%** lidí.



Obrázek 15: Graf určující systém algoritmu pro simulaci virtuálních klientů v systému

U každé hrany je také napsaný popis, co má daný klient provádět v uzlu, do kterého daná hrana směřuje. Popišme si tedy jednotlivé uzly.

- **1** - Vstupní uzel, kde se zobrazí celkový rozvrh na přístroje v rámci daného dne a možnost přihlášení se jako náhradník na příslušný přístroj a čas. Z tohoto místa **75%** virtuálních uživatelů provede rezervaci na přístroj (hrana do uzlu **2**), **10%** načte znovu daný rozvrh , a to z důvodu,že například přejde na jiný den a až tam se rezervuje, a dalších **10%** se nahlásí jako náhradník na příslušný přístroj. Zbylých **5%** jsou uživatelé, kteří se jen podívali na rozvrh, a zjistili, že pro ně neexistují volná místa pro rezervaci a odejdou pryč ze systému, tudíž jsou odkázání odejít do koncového uzlu **13**.

- **2** - Prostor kde se provádí rezervace na přístroje v určitý den a hodinu. Zde se vyskytuje smyčka, která je využita pro opakované rezervování se na různé přístroje v daný den s pravděpodobností **40%**. Pokud bychom se chtěli rezervovat na jiný den, musíme opět přejít do uzlu **1** pro načtení rozvrhu jiného dne (využije jen **15%** virt. uživatelů), a poté znovu zpět do uzlu **2** k vytvoření další rezervace. Z daného uzlu **2** jsou ještě další 4 cesty, a to do uzlu **4** (**15%**), který nám zobrazí seznam všech rezervací na přístroje, do uzlu **11** (**2%**) pro kontrolu operací, že nám rezervovaná hodina opravdu odebrala tolik kreditů, kolik měla, dále do uzlu **12** (**2%**), kde se klient může podívat na počet svých rezervací a odebraných kreditů za určité období a nakonec do koncového uzlu **13(26%)**, kdy vlastně klient v předchozím kroku provedl příslušnou rezervaci a následně odejde ze systému.

- **3** - Nahlášení se jako náhradník na příslušný přístroj a hodinu. Tato operace se provádí v prostředí, kde je stále vidět rozvrh hodin, a proto je možné se z tohoto místa ještě rezervovat. **50%** virtuálních uživatelů ještě využije možnost dané rezervace.

- **4** - Vstupní uzel, který reprezentuje zobrazení všech rezervovaných přístrojů v příslušných dnech a časech. Po příchodu do této oblasti a načtení rezervací se uživatel může rozhodnout, že provede ještě nějaké další rezervace, případně, že chce nějaké smazat. Pro nové rezervace musí jít opět na seznam rezervací do uzlu 1 (tuto operaci využije **30%** daných uživatelů) a pro smazání rezervací do uzlu **5** (**30%**).

- **5** - Tento uzel reprezentuje smazání příslušné rezervace. Uživatel může toto mazání opakovat, proto z tohoto uzlu vede hrana zpět do uzlu **4**(**30%**) na seznam rezervací. Navíc, když uživatel nějakou rezervaci smaže, přičtou se mu zpět kredity, které za ni utratil a tím pádem má možnost dalších rezervací. Proto se může dostat zpět do uzlu **1** s pravděpodobnosti **30%**.

- **6** - Opět vstupní uzel, který tentokrát zaujímá roli pro zobrazení týdenního rozvrhu pro rezervace na cvičení, a také jako v uzlu **1** je zde možnost nahlásit se jako náhradník. Systém odchozích hran z tohoto uzlu je analogický jako z uzlu **1**. Pouze se jedná o hrany vedoucích do uzlů reprezentujících operace s cvičeními, a ne s přístroji. Analogické jsou i procentuální ohodnocení daných hran.

- **7** - Zde se může klient nahlásit jako náhradník na příslušné cvičení v určitý den a hodinu.
- **8** - Uzel, který má stejnou funkčnost a analogii jako uzel **2** jen s tím rozdílem, že se jedná o rezervace na cvičení.
- **9** - Vstupní uzel pro zobrazení seznamu rezervací na cvičení v určitých dnech a časech. Opět analogie s uzlem **4**.
- **10** - Smazání rezervací na cvičení (analogie s uzlem **5**).
- **11** - Vstupní uzel do statistiky, kde klienti vidí počet svých rezervací, jak celkových tak konkrétních na daný přístroj a cvičení v určitém období. Dále počet celkových vložených a odečtených kreditů rezervacemi v rámci určitého období, a to opět jak celkových, tak konkrétních na určité přístroje a cvičení. Z tohoto uzlu je možné se dostat do všech vstupních uzlů, tedy do **1**, **4**, **6**, **9** s celkovou pravděpodobností **40%** (každý uzel **10%**). Zbylých **60%** putuje do koncového uzlu.
- **12** - Vstupní uzel pro archivu, kde klienti vidí, jaké operace prováděli co se přidávání a odebírání rezervací a kreditů týče, případně jaké operace prováděli administrátoři s jejich rezervacemi a kredity. Zde je opět možnost, jako v uzlu **11**, přejít do všech vstupních uzlů **1**, **4**, **6**, **9** s analogickou procentuální pravděpodobnosti.

Toto znázornění veškerých uzlů a cest v systému nám velice usnadní následné naprogramování daného průchodu.

Ještě bych upřesnil funkcionalitu jednotlivých uzlů. Pokud bychom tedy vzali v úvahu uzly **6 - 10** jako první komponentu daného grafu a uzly **1 - 5** jako druhou komponentu a obě tyto komponenty od grafu oddělili, budou navzájem izomorfní. Tzn. uzel **1** je izomorfní s uzlem **6**, uzel **2** s uzlem **8** apod. Což nám podstatně usnadní programování daného průchodu grafem v tom smyslu, že pro uzel **1** a **6** bude stačit pouze jedna funkce, do které budeme posílat jen jeden parametr, který bude určovat, zda se jedná o přístroje, či cvičení.

8.2.2 Naprogramování virtuálního průchodu systémem

V každém uzlu se provádějí určité operace, které jsou pro daný uzel typické, neprovádějí se nikde jinde a navíc se tyto operace při každém vstupu do uzlu opakují. Proto, co se programování týče, tyto uzly nahradíme funkcemi (metodami) obsahujícími zadané instrukce vzhledem k danému uzlu a pomocí ukazatelů na tyto funkce se mezi nimi budeme pohybovat. Instrukcemi v každé funkci jsou jednotlivé SQL dotazy, které bylo třeba připravit a dohledat v systému. Jsou to ty samé SQL dotazy, které se využívají ke klasické činnosti v systému.

Samozřejmě že ale nešlo použít jen samotné dotazy, bylo třeba připravit další související podklady, a to například, který přístroj, či cvičení se rezervuje, který klient, v

který den a dobu tyto rezervace vypisuje, maže apod. Tyto další podklady se potom také přenášely mezi funkcemi, aby dotazy správně fungovaly a vkládaly, editovaly a mazaly data tak, jako by to dělal běžný klient.

Další nutností ještě bylo naplnit systémovou databázi daty, které se využívaly v průběhu průchodu systémem. Tato data mi byla poskytnuta z jednoho brněnského fitcentra, kde mi bylo dodáno 4 000 uživatelů a více než 90 000 rezervací jak na přístroje, tak na cvičení. Díky těmto datům bylo možné provést analýzu SQL dotazů a následně jejich odladění. Více v kapitole (7.2). Jen ještě zdůrazním, že dodaná data jsou důvěrná, smlouvou s fitcentrem zajištěná a nemám právo je nijak zveřejňovat, či dále poskytovat třetí osobě, aby nedošlo k jejím zneužití.

Nakonec bylo třeba ve funkcích vyřešit způsob přechodu z jedné funkce do druhé v závislosti na procentuálnímu ohodnocení hrany mezi nimi. Pro řešení tohoto přechodu připadly v úvahu dvě možnosti, a to

- pomocí náhody s jednoduchým vygenerováním náhodného čísla viz obrázek (16) nebo
- nastavit pevně danou cestu systémem.

Obrázek 16: Určení náhodného bodu v procentuálně rozložené oblasti

Aby testování vytíženosti bylo správně odladěné, bylo nutné vytvořit pevně zadanou cestu systémem, a to proto, že pokud bychom využili náhodu, průchod by sice vystihoval realističtější chování klientů v systému než pomocí pevně zadané cesty, ale při testování vytíženosti by vždy ukázal jiné hodnoty. Nepozanali bychom jestli jsme úpravou SQL dotazů systém zlepšili nebo ne.

Přechod jsem tedy vyřešil tak, že jsem přesně určil, kteří uživatelé půjdou kterou cestou v závisosti na procentuálním ohodnocení. Například, prvních x lidí půjde první cestou, druhých x lidí druhou apod.

Tento systém putování opakujeme v každém uzlu, dokud veškeří virtuální klienti neodejdou ze systému. Pro testování a odladění jsem nechal do systému vstoupit 1000 uživatelů a výsledky analýzy jednotlivých kritických SQL dotazů si ukážeme v dalších kapitolách.

8.2.3 Odladění fyzického návrhu databáze

Jak jsem již psal výše, SQL Server obsahuje určité nástroje pro určování kritických SQL dotazů, které zpomalují celkový chod systému a jejich doladění. Díky nim můžeme sledovat konkrétní vlastnosti dotazů, na základě kterých potom rozhodneme, zda dotaz doladit, upravit jeho strukturu příkazu či ne.

Vlastnostmi dotazů rozumíme konkrétně tyto:

- **READ** - tato vlastnost se v SQL Serveru dělí na logické (logical read) a fyzické (physical read) načítání datových stránek. Fyzická část načítá danou stránku do paměti (buffer) z disku, zatímco logická načítá danou stránku z paměti. V našem případě budeme testovat logical read, tedy hodnoty určující počet stránek, které bylo potřeba načíst k vykonání daného dotazu.
- **WRITE** - je počet stránek zapsaných do paměti při provádění daného dotazu,
- **DURATION** - celkový čas, který byl zapotřebí k provedení příslušného dotazu (většinou uváděný v milisekundách),
- **CPU** - čas strávený na CPU v rámci vykonáváno dotazu (opět uváděný v milisekundách).

Jsou to nejdůležitější vlastnosti, které SQL Server nabízí a u každé z nich se budeme snažit zredukovat jejich hodnotu. Čím menší hodnota, tím lepší. I když existují i výjimky.

Mezi nástroje SQL Serveru, které budeme využívat pro zachycení jednotlivých dotazů a jejich analýzu patří

- **SQL Server Profiler** - díky němu zachytíme veškeré prováděné operace, zjistíme hodnoty jednotlivých vlastností dotazů a uložíme tato data do souboru. Veškeré tyto operace provedeme na straně klienta. Nejdříve tedy v našem systému musíme spustit naprogramovanou virtuání „procházku“, spustit daný nástroj a ihned uvidíme jak SQL Server odchytává dané dotazy v Profileru. Po ukončení procesu se v okně nachází velké množství řádků operací a my můžeme zkoumat dané vlastnosti dotazů. Vidíme ihned, jak dlouho trvalo vykonání daného dotazu, kolik datových stránek bylo načteno (read) a kolik uloženo (write), nicméně tato soupiska pořád není tak přehledná, abychom rychle našli ty nejhorší dotazy, které bychom chtěli odladit a provést hlubší analýzu. Pro takovouto analýzu je lepší využít nástrojů RML[9].
- **Execution Plan** - nástroj pro zkoumání konkrétního dotazu, jeho konkrétních částí a zkoumání jak se daný dotaz chová, jak vyhledává data v databázi a jak dané tabulky spojuje. Narozdíl od Profileru, je jeho znázornění určitých výsledku v podobě grafu obsahující cesty mezi jednotlivými vyhodnocenými částmi dotazu.

[9] http://support.microsoft.com/kb/944837/en-us?fr=1

Mezi zmiňované RML nástroje, které nám usnadní práci se zachycenými daty Profilerem patří

- **ReadTrace** - nástroj, který uložená data z Profileru načte, zpracuje a připraví nám je k prohlédnutí.
- **Reporter** - díky tomuto nástroji připravená data můžeme jednoduše analyzovat, poskytuje nám různé typy grafů a podrobných analýz jak jednotlivých konkrétních dotazů, tak celkového průběhu „procházky" systémem.

8.2.3.1 Odladění konkrétních dotazů a nástroj Execution plan

První vlastnost, kterou jsem na dotazech zkoumal byla *DURATION*, tedy celková doba jeho vykonání. Po prohlédnutí všech unikátních dotazů v nástroji **Reporter** jsem zjistil, že maximální doba provedení jednoho z nich byla 1 sekunda (**„dotaz 1"**). Našel jsem ještě jeden, který trval 200ms, a také ho zařadil na soupisku (**„dotaz 2"**). Další dotazy už nepřekročily hodnotu větší jak 30ms, takže jsem v tomto případě skončil.

Po prohlédnutí vlastnosti *CPU*, že se chová analogicky jako *DURATION*, tudíž se seznam dotazů k ladění nijak nezvětšil.

Co se týče vlastnosti *WRITE*, nenabývala žádných vysokých hodnot, v systému se spíše vypisují seznamy rezervací, rozvrhy hodin apod. Jediný zápis probíhá při zarezervování se a tudíž tyto hodnoty nebyly až tak vysoké. Maximální hodnota byla 49.

Poslední vlastnost, podle mého názoru v tomto případě ta nejdůležitější, kterou bylo třeba prozkoumat, je počet načtených datových stránek *READ*. V systému se pořád něco zobrazuje, ať se podíváte do jakékoliv části, vždy se načítá rozvrh, případně seznam různých rezervací, které jsou zkombinované s výpisem jednotlivých přístroju a cvičení, k tomu přiřazeni cvičitelé a doby kdy se mají daná cvičení zobrazit, toto vše zabírá spousty datových stránek. Maximální počet *READ* hodnot obsahuje už zmiňovaný **„dotaz 1"**. Za ním následuje **„dotaz 2"** a po pečlivém prozkoumání jsem narazil ještě na další 3 dotazy, jejichž hodnoty *READ* jsou větší než 100. Nižší hodnotu jsem už nebral v úvahu.

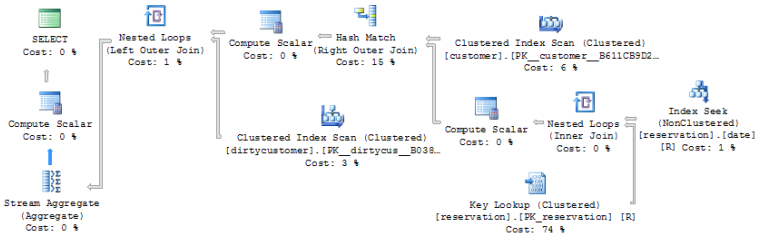
První dotaz tedy je (**„dotaz 1"**)

```
SELECT C.login,R.reservationID,R.combination,CAST(R.note AS VARCHAR(8000)) as resNote,R.
    trainer,R.date,R.equipmentID,R.customerID,C.credit,C.email,C.tel,C.adminAdd,CAST(C.notes
    AS VARCHAR(8000))as notes,Count(DC.customerID) as 'countD'

FROM customer C RIGHT JOIN reservation R ON R.customerID = C.customerID LEFT JOIN
    dirtycustomer DC ON (DC.customerID = R.customerID AND DC.deleted = 0)

WHERE R.date > '2011-10-25' AND R.date < '2011-10-25 23:59:59' AND R.deleted = 0
GROUP BY R.date,R.equipmentID,C.login,R.reservationID,CAST(R.note AS VARCHAR(8000)),R.
    combination,R.trainer,R.customerID,C.credit,C.email,C.tel,C.adminAdd,CAST(C.notes AS
    VARCHAR(8000))
```

Výpis 4: První dotaz k ladění

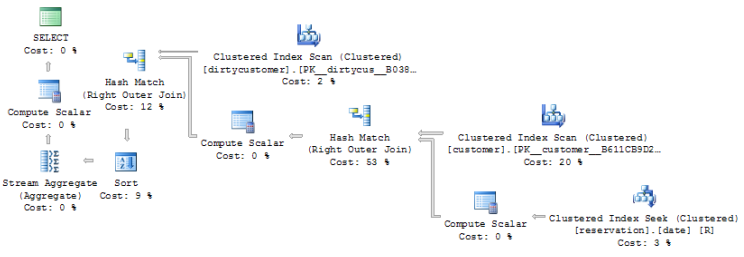
a jeho účelem je zobrazení rozvrhu na přístroje na konkrétní den, vypsání jednotlivých rezervací, jméno uživatele, který danou rezervaci vlastní, a také jestli obsahuje nějaké prošlé rezervace (dirtycustomer).

K tomu, abychom mohli lépe určit, co na dotazu vylepšit, která jeho část způsobuje nárust hodnot *READ* využijeme **Execution plan**, který nalezneme přímo v SQL Server Management studiu. Na obrázku (17) vidíme ocenění jednotlivých části dotazu a my se budeme soustředit na ty části, které vykazují vysoké hodnoty **Cost**.



Obrázek 17: Execution plán prvního ladícího dotazu

Nepřehlédnutelnou hodnotou je 74% u části, kde je zobrazen primární index **PK_reservation**. Dále je tato položka označena jako **clustered**, což znamená, že data v tabulce jsou seřazena podle tohoto atributu. Jelikož tabulka obsahující tento primární index obsahuje více než 40000 záznamů, je ocenění této části obrovské. Navíc když vidíme v dotazu, že vybíráme data z tabulky na základě atributu **date**, což je také index. Čili pokud bychom měli data v tabulce seřazená podle tohoto data, byl by možná tento dotaz lepší. Měl by hned přístup ke všem datům pohromadě. Nyní musí projet celou databázi a hledat dané záznamy v celé tabulce. Zkusíme tedy změnit clustered index (primírního indexu) na non-clustered a atribut date změníme na clustered a uvidíme jak se daný dotaz zlepší.

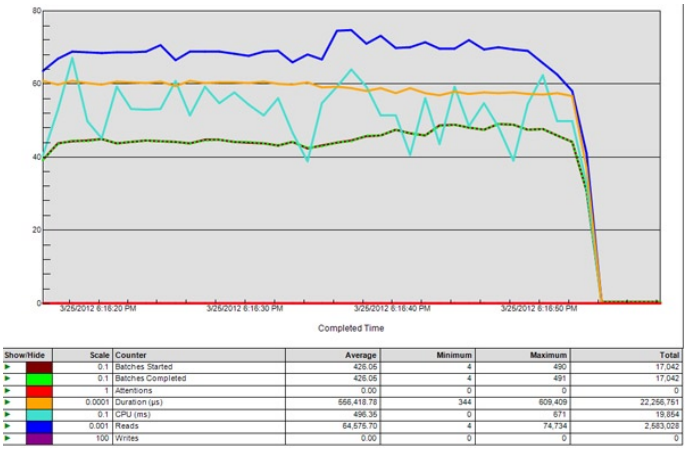


Obrázek 18: Execution plán prvního přeladěného dotazu

Pokud se nyní koukneme na obrázek (18), uvidíme, že ohodnocení jednotlivých částí dotazu se změnila, úplně zmizela část s indexem **PK_reservation**, protože vlastně v dotaze není jeho potřeba, nicméně se objevila nová část, a to část reprezentující JOIN mezi tabulkou reservation a customer. Jelikož ale uživatel obsahuje hodně rezervací, toto

spojení je nutné pro dotaz a můžeme znovu zkusit spustit průchod systémem, jestli se hodnoty vlastností SQL dotazů zmenší.

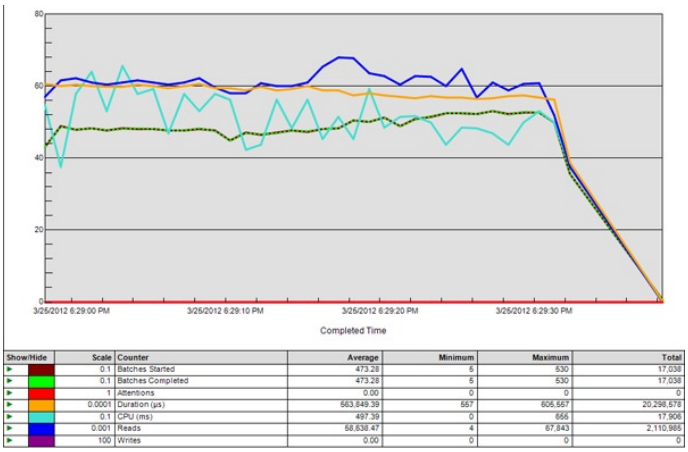
Opět jsem pustil do systému 1000 klientů a zde už se objevily rozdíly hlavně v *READ* vlastnosti. V původní neupravené databázi bylo potřeba k celému průchodu systému načíst 2 583 028 datových stránek, tak jak můžeme vidět na obrázku (19). Celkový průběh trval 22 sekund a nejdéle trval dotaz 671 ms.



Show/Hide	Scale	Counter	Average	Minimum	Maximum	Total
▶	0.1	Batches Started	426.05	4	490	17,042
▶	0.1	Batches Completed	426.05	4	491	17,042
▶	1	Attentions	0.00	0	0	0
▶	0.0001	Duration (µs)	566,418.78	344	609,409	22,256,751
▶	0.1	CPU (ms)	496.35	0	671	19,854
▶	0.001	Reads	64,675.70	4	74,734	2,583,028
▶	100	Writes	0.00	0	0	0

Obrázek 19: Reporter - graf průchodu neupravené databáze

Po úpravě indexu v atributu date se počet datových stránek celkového průchodu snížil na 2 110 985 viz obrázek (20), což je o pětinu lepší výsledek než v neupravené databázi. Také celkový průběh se zrychlil o 2 sekundy a nejdéle trval dotaz 655 ms. Co se rychlosti týče, tak moc razantní změny nenastaly narozdíl od vlastnosti *READ*.



Show/Hide	Scale	Counter	Average	Minimum	Maximum	Total
▶	0.1	Batches Started	473.28	5	530	17,038
▶	0.1	Batches Completed	473.28	5	530	17,038
▶	1	Attentions	0.00	0	0	0
▶	0.0001	Duration (µs)	583,849.39	557	605,557	20,298,578
▶	0.1	CPU (ms)	497.39	0	555	17,906
▶	0.001	Reads	58,838.47	4	67,843	2,110,986
▶	100	Writes	0.00	0	0	0

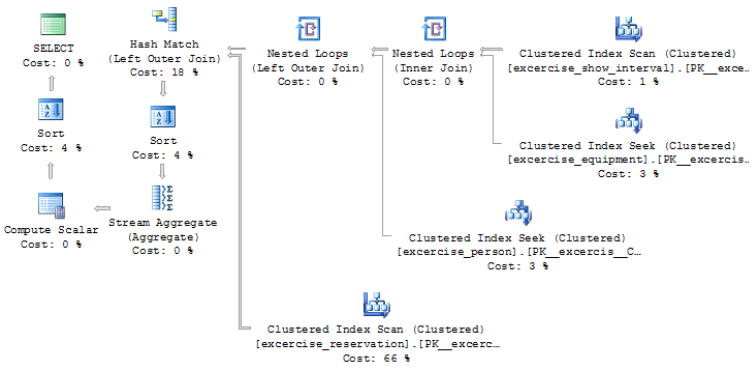
Obrázek 20: Reporter - graf průchodu po úpravě tabulky pro rezervace na přístroje

Podívejme se nyní na druhý dotaz, který využívá jinou část systému, a to zobrazení rozvrhu rezervací na cvičení. Pokud se podaří zlepšit i tuto část, celkový systém se zase posune o něco k lepšímu. Jedná se tedy o dotaz

```
SELECT E.fromT,E.toT,E.title,E.day,P.title  as  'name',P.color,E.excerciseEquipmentID,E.capacity,E
    .price,count(ER.excerciseReservationID) AS reservedCapacity

FROM excercise_equipment E
    JOIN excercise_show_interval ESI   ON (ESI.excerciseEquipmentID = E.excerciseEquipmentID
        AND ESI.deleted = '0')
    LEFT JOIN excercise_person P           ON (P.excercisePersonID = E.excercisePersonID)
    LEFT JOIN excercise_reservation ER  ON (E.excerciseEquipmentID = ER.
        excerciseEquipmentID AND '2011-12-12' <= ER.date AND ER.date <= '2011-12-18
        23:59:59' AND ER.hallID = '1' AND ER.deleted = '0')

WHERE 1=1 AND E.hallID = '1' AND E.deleted = 0
    AND (
        (ESI.dateFrom >= '2011-12-12' AND '2011-12-18' >= ESI.dateTo AND ESI.dateTo <> '
            1900-01-01') OR
        (ESI.dateFrom <= '2011-12-18' AND '2011-12-18' <= ESI.dateTo AND ESI.dateTo <> '
            1900-01-01') OR
        (ESI.dateFrom <= '2011-12-12' AND '2011-12-12' <= ESI.dateTo AND ESI.dateTo <> '
            1900-01-01') OR
        (ESI.dateFrom <= '2011-12-12' AND '2011-12-18' <= ESI.dateTo AND ESI.dateTo <> '
            1900-01-01') OR
        (ESI.dateFrom <= '2011-12-18' AND '1900-01-01' = ESI.dateTo AND ESI.dateFrom <>
            '1900-01-01')
    )
GROUP BY E.excerciseEquipmentID,E.fromT,E.toT,E.title,E.day,P.title,P.color,E.capacity,E.price
ORDER BY E.day ASC
```

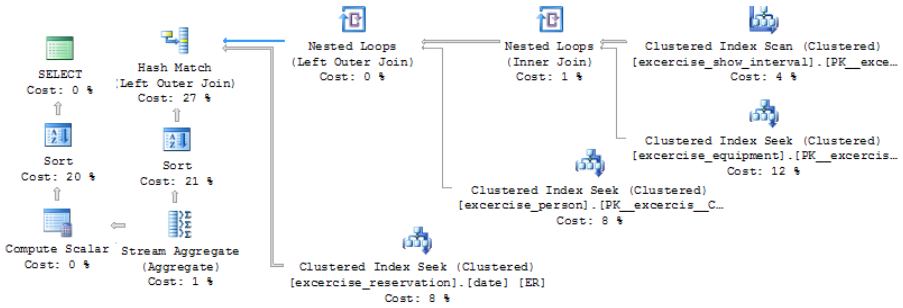
Výpis 5: Druhý dotaz k ladění

a na obrázku (21) si opět ukažme **Execution plan** na kterém si ukážeme kritické části dotazu.



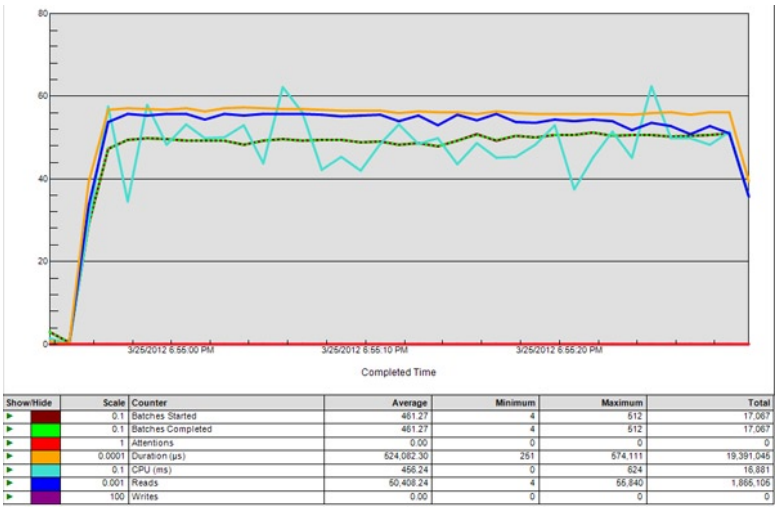
Obrázek 21: Execution plán druhého dotazu

Zde se nám nabízí stejná možnost jako v předchozím dotazu, kde primární index **excercise_reservation** můžeme opět změnit z **clustered** na **non-clustered** a index date nastavit na **clustered**. Když jsem toto v databázi provedl, jak je vidět na obrázku (22), opět se hodnoty části v plánu změnily a zmizela část s primárním klíčem. Objevil se menší nárust části pro JOIN, to je ale třeba pro spojení rezervací s jednotlivými cvičeními.



Obrázek 22: Execution plán druhého vyladěného dotazu

Po provedení a prozkoumání vlastností se opět snížila hodnota *READ*, a to na 1 865 108 datových stránek, tak jak je vidět na obrázku (23), čímž jsme vlastně už teď snížili systému hodnotu *READ* o 700 tisíc datových stránek. Celkový průběh se také zrychlil, a to o sekundu a maximální doba provedení jednoho dotazu klesla na 624 ms, což je jen o nějakých 30 ms. Opět žádné závratné zlepšení co se času týče, nicméně celkový počet datových stránek se výrazně snížil.



Show/Hide	Scale	Counter	Average	Minimum	Maximum	Total
▸	0.1	Batches Started	461.27	4	512	17,067
▸	0.1	Batches Completed	461.27	4	512	17,067
▸	1	Attentions	0.00	0	0	0
▸	0.0001	Duration (µs)	524,082.30	251	574,111	19,391,046
▸	0.1	CPU (ms)	456.24	0	624	16,881
▸	0.001	Reads	50,408.24	4	55,840	1,865,106
▸	100	Writes	0.00	0	0	0

Obrázek 23: Reporter - graf průchodu po úpravě tabulky pro rezervace na cvičení

Zbývají nám ještě 3 dotazy, které už nejsou tak obsáhlé jako předchozí dva, proto už nebudu ukazovat jejich **Execution plan**. Jedná se o dotazy

```
SELECT *
FROM reservation
WHERE (date > '2011-3-13' AND date < '2011-3-13 23:59:59' AND deleted = 0 )
ORDER BY equipmentID ASC
```

Výpis 6: Třetí dotaz k ladění

```
SELECT SUM(R.price) as 'creditCount',SUM(R.bonus) as 'bonusCount', COUNT(R.
    reservationID) as 'resCount', E.title
FROM equipment E LEFT JOIN reservation R ON (E.equipmentID = R.equipmentID AND date >=
    '2011-12-01' AND date <= '2011-12-01 23:59:59' AND customerID = 3503 AND (
    deleted = '1' OR deleted = '0') AND combination = 0)

GROUP BY E.equipmentID, E.title,E.priority
ORDER BY E.priority
```

Výpis 7: Čtvrtý dotaz k ladění

```
SELECT *
FROM excercise_reservation
WHERE excerciseEquipmentID = 17 AND deleted = 0 AND date = '2011-11-02'
```

Výpis 8: Pátý dotaz k ladění

Ve všech posledních třech dotazech se opět objevuje atribut **date**, který jsem ladil v prvních dvou dotazech, takže zde asi už nebude co upravovat. Nicméně se podařilo i tak vylepšit celkové vytížení systému téměř o třetinu, což je, podle mého názoru, dobré zlepšení a určitě dopomůže lepšímu chodu.

8.3 Audit databázových operací

Poslední částí kapitoly ohledně rozšíření pro SQL Server je audit databázových operací. Jedná se o nastavení SQL Serveru, který bude zachytávat veškerou činnost administrátorů, kteří se serverem pracují a provádějí v něm určité SQL dotazy. Toto zachytávání činnosti je velice dobré k tomu, že máte kontrolu nad tím, kdo co prováděl a případně tyto lidi za jejich úkony potrestat, když by provedli nějaké špatné nastavení či operaci s databází.

Audit si můžete naprogramovat ve své aplikaci i sami, sami si můžete vytvořit funkce, které vždy uloží do databáze tu činnost, kterou jste zrovna provedli, nicméně SQL Server už toto vše má v sobě, proto využijeme jeho nástrojů a vytvoříme audit v něm. Má v sobě už několik předdefinovaných aktivit k provádění auditu a navíc můžete provádět audit auditu. Výsledky zachytávání aktivit jsou ukládány na disku nebo logu operačního systému. Výhoda dat na disku je ta, že je můžete znovu nahrát do databáze pro budoucí analýzu. Příkazy jako **SELECT**,**UPDATE**, **DELETE**,**INSERT** mohou být zvlášť zachytávány pro konkrétní uživatelé daného serveru, pro uživatele 1 nastavíte například

jen příkaz *INSERT*, pro uživatele 2 naříklad jen *SELECT*. Záleží, kdo a co bude provádět. Veškerá täto nastavení a vytvoření auditu se dají provádět přímo v nástroji SQL Serveru nazvaného SQL Server Management.

Prvním krokem k jeho vytvoření je třeba vytvořit **SQL Server Audit object**, přiřadíte mu určité jméno, vyberete příslušná nastavení a nastavíte cestu kam ukládat výsledky. Dané nastavení uložíte a nyní, jako druhý krok, je třeba vytvořit **Audit Specification**. SQL Server obsahuje 2 typy **Audit Specification**, a to

- **Server Audit Specifications** - využívá se, když chcete provádět audit na serveru, například přihlášení a odhlášení uživatele do databáze.
- **Database Audit Specifications** - ten se využívá, když chcete monitorovat veškeré pohyby v databázi, jako například, kdo provádí SELECT příkaz na určitých datech apod.

Každý z těchto typů se potom přiřadí k **SQL Server Audit object**. Tím pádem Server a Database Audit Specifications se vytváří separátně. Nemohou být oba dva zároveň vytvořeny na jednom **SQL Server Audit object**, musí se vytvořit dva různé. Jakmile máte oba dva audity vytvořeny, je třeba je ještě zapnout a od této chvíle budou zaznamenávat vše, co jste jim nastavili. Výsledky se potom dají, pokud je ukládáte do souboru, importovat do databáze a sledovat co vše obsahují, případně pokud data ukládáte do logu, tak pro zobrazení dat můžete využít Event log Reader.

8.3.1 Konkrétní využití

Pro využití auditu v mém systému jsem použil **Database Audit Specifications**, a to z toho důvodu, že chci zachytávat veškeré operace přihlášených uživatelů do konkrétní databáze. Chci, abych měl kontrolu nad tím, jestli upravovali nějaké tabulky, jestli prováděli operace s daty, například s jednotlivými rezervacemi, nebo kredity. Rezervace a kredity jsou totiž velmi citlivá data a pokud by se stala nějaká chyba, určité fitcentrum potom může začít i prodělávat, proto je důležité mít přehled nad veškerými operacemi.

Při vytváření jsem tedy vytvořil 4 typy auditu, a to **SELECT**, **UPDATE**, **DELETE** a **INSERT** a ke každému z nich jsem přiřadil jen ty nejdůležitější tabulky, které bych chtěl zkoumat. Není dobré přidávat veškeré tabulky systému, protože tyto operace také zpomalují SQL Server, a proto vyberu jen ty nejdůležitější. Jak už jsem psal, jedná se o tabulky pro uchování veškerých dat ohledně rezervací na přístroje a na cvičení, tabulky ohledně kreditového systému a v neposlední řadě tabulka se všemi klienty. Dále bylo ještě třeba zvolit typ uživatelů, které audit bude zkoumat. Zvolil jsem typ **Public** abych tím pokryl veškeré typy možných přihlášených uživatelů v SQL Serveru.

Jak už jsem psal výše, audit v SQL Serveru umožňuje ukládat data třemi možnými způsoby, a to

- **File** - data budou uložena do souboru.

- **Security log** - data jsou uložena v příslušném security logu operačního systému.
- **Application log** - data jsou uložena v příslušném application logu operačního systému.

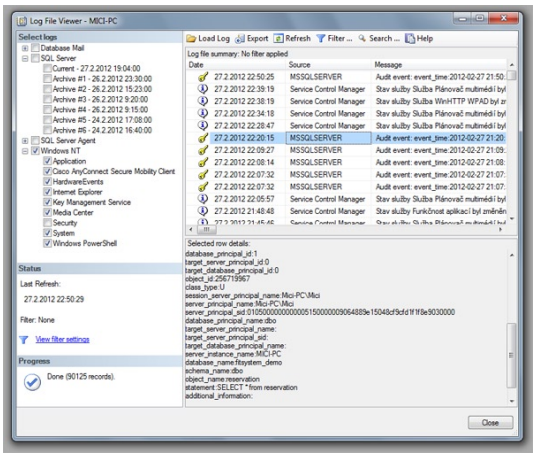
V rámci této diplomové práce jsem zvolil třetí variantu, a to z toho důvodu, že data potom můžeme jednoduše zobrazit přímo v SQL Server Management Studiu.

8.3.2 Zobrazení výsledků auditu

K tomu, abychom mohli výsledky zobrazit, je třeba nasimulovat chování přihlášených uživatelů k SQL Serveru. Provdeme tedy určité SQL dotazy pro výběr, nebo například pro vložení dat do databáze a můžeme zkontrolovat, jestli audit dané operace zachytil.

Zobrazení výsledků můžeme provést opět v nástrojích SQL Serveru, a to konkrétně v **Log File Viewer**. Tento nástroj spustíme v levé stromové struktuře, konkrétně ve složce **Management** ⇒ **SQL Serve logs**. Pokud na ni klikneme pravým tlačítkem a dáme vybrat **SQL Server a Windows log**, zobrazí se nám seznam všech operací, které byly zachyceny auditem. Po vyskočení nového dialogu jsou automaticky v levé části zaškrtnuty možnosti Windows NT a SQL Server. Pro nás je důležitá první zmíněná možnost, a to z toho důvodu, že ukládáme data do **Application Log**.

V pravé části dialogu potom vidíme jednotlivé záznamy, které se ukládaly. Jsou označeny příznakem **MSSQLSERVER**, tak jak je zobrazeno na obrázku (24) a pokud na některý z nich klikneme, ve spodní části se nám ukáží jednotlivé vlastnosti daného logu, zejména dotaz, který byl prováděn, která tabulka byla byla využita, kdo tento dotaz provedl apod.



Obrázek 24: Výpis jednotlivých záznamu uložených v rámci auditu

9 Závěr

Cílem této diplomové práce bylo naprogramovat několik nových modulů rezervačního systému Fitsystém a dále pak rozšířit datovou vrstvu systému pro SQL Server, provést ladění nejnáročnějších SQL dotazů a nastavit audit.

Co se modulů týče, v průběhu psaní této diplomové práce si systém objednali další tři noví klienti a nabídka nových modulů byla jednou z věcí, která rozhodla, že si tento systém objednali, že se nerozhodli pro konkurenci. Největší úspěch měl kreditový systém a statistika. Naopak nejmenší úspěch měl modul synchronizace s google kalendářem, který si neobjednal vůbec nikdo. Nicméně i tak zvolené moduly přispěly k lepší propagaci systému a k jeho robustnosti. Díky těmto modulům lze oslovit větší část klientů, kteří potřebují pro své podnikání určitý typ rezervací. Systém se stal více obecným nástrojem.

Co se převodu na SQL Server týče, povedlo se převést kompletně celý rezervační systém, veškeré jeho části a moduly správně fungují, takže pokud by klient vyžadoval provoz na tomto serveru, nebylo by to překážkou. Dále se mi povedlo Fitsystém vylepšit v rámci ladění SQL dotazů nad databází, a to konkrétně v jejich vlastnosti READ. Tuto hodnotu se povedlo snížit skoro o třetinu oproti původním hodnotám, což, si myslím, je dostatečné zlepšení. A nakonec nastavení auditu databázových operací proběhlo také v pořádku, bylo nastaveno sledování nejdůležitějších příkazů v rámci nejcitlivějších tabulek, takže pokud by klient tento audit vyžadoval, jeho nastavení by nebylo žádným problémem.

10 Literatura

[1] **Různé typy single sign-on protokolů.**
http://en.wikipedia.org/wiki/List_of_single_sign-on_implementations

[2] **Oficiální stránky kerbera na MIT.**
http://web.mit.edu/kerberos/

[3] **Kerberos pro Windows.**
http://technet.microsoft.com/en-us/library/cc753173%28WS.10%29.aspx

[4] **MyOpenID služba.**
https://www.myopenid.com/

[5] **MojeID služba.**
http://www.mojeid.cz

[6] **Microsoft podporuje OpenID.**
http://openid.net/tag/microsoft/

[7] **OAuth protokol - oficiální stránka.**
http://http://www.oauth.net/

[8] **Hybridní protokol kombinující OAuth a OpenID.**
http://code.google.com/p/step2/

[9] **3D Secure systém od České spořitelny.**
http://www.csas.cz/banka/content/inet/internet/cs/sc_1585.xml

[10] **Průběh platby na PayU**
http://www.payu.cz/prubeh-platby-pres-payu,1024.html

A Datový slovník

A.1 Permanentky

A.1.1 Tabulka permanent_pass

- **permanentPassID** - jednoznačné určení dané permanentky,
- **name** - název permanentky,
- **count** - pro kolik vstupů je daná permanentka vytvořena,
- **dateAdd** - datum vytvoření permanentky,
- **dateEdit** - datum editace permanentky,
- **userAdd** - kdo danou permanentku vytořil,
- **userEdit** - kdo danou permanentku smazal,
- **deleted** - jestli je daná permanentka smazána, či ne (obsahuje hodnoty 0 a 1),
- **dateDeleted** - kdy byla daná permanentka smazána.

A.1.2 Tabulka customer_permanent_pass

- **customerPermanentPassID** - jednoznačné určení řádku tabulky,
- **customerID** - ID daného uživatele, kterému je vkládána permanentka,
- **permanentID** - ID vložené permanentky,
- **dateTo** - platnost permanentky u daného uživatele,
- **count** - kolikještě permanentka obsahuje vstupů (mění se v závislosti na počtu rezervací),
- **dateAdd** - datum vložení permanentky danému klientovi,
- **userAdd** - kdo danou permanentku vložil klientovi,
- **deleted** - jestli je daná permanentka smazána, či ne (obsahuje hodnoty 0 a 1),
- **dateDeleted** - kdy byla daná permanentka smazána.

A.2 Kreditový systém

A.2.1 Obchod

- **shopID** - jednoznačné určení produktu,
- **name** - jméno položky,
- **credit** - počet kreditů, které daná položka odebere při nákupu,
- **dateAdd** - datum vytvoření produktu,
- **dateEdit** - datum editace produktu,
- **userAdd** - kdo daný produktu vytořil,
- **userEdit** - kdo daný produktu smazal,

A.3 Platební modul PayU

A.3.1 Tabulka payu_payment

- **payuPaymentID** - jednoznačné určení platby,
- **customerID** - ID uživatele, který platbu prováděl,
- **posID** - kód přidělený od PayU aby poznali, že se jedná o nás,
- **posAuthKey** - druhý kód přidělený od PayU aby poznali, že se jedná o nás,
- **sessionID** - jednoznačné určení transakce pro PayU,
- **payType** - typ platby,
- **amount** - částka, kterou je třeba zaplatit (uvádí se v haléřích),
- **description** - popis platby, za co se zaplatilo apod.,
- **clientIP** - IP adresa klienta, který transakci provedl,
- **dateAdd** - datum provedení transakce,
- **dateEdit** - datum změny transakce (například při změně statusu),
- **transactionStatus** - v jakém stavu se daná transakce nachází, jestli provedená, neuspěšná, čekající apod.,
- **asignMoney** - Jestli se budou peníze uživateli přidávat. Může nastat, že se platba zruší, ale dokončí, proto musíme zadat tomuto parametru hodnotu 0 když se například zruší

A.3.2 Tabulka payu_info

- **payuInfoID** - jednoznačné určení daného oznámení,
- **payuPaymentID** - ID platby pro kterou dané oznámení má být,
- **customerID** - ID uživatele, kterému oznámení patří,
- **transID** - identifikátor transakce vytvořený v PayU,
- **posID** - kód přidělený od PayU aby poznali, že se jedná o nás,
- **payType** - typ platby,
- **amountPS** - ,
- **successOrError** - jestli byla transakce úspěšná nebo ne,
- **error** - typ erroru (501,206, apod.),
- **errorString** - Zpráva erroru,
- **date** - datum oznámení

A.4 Statistika

A.4.1 Tabulka statistic_day_count

- **statisticDayCountID** - jednoznačné určení řádku tabulky,
- **date** - den ve kterém se dané rezervace prováděly,
- **resCount** - celkový počet rezervací jak přístrojů, cvičení tak produktů v rámci dne **date**,
- **resRemoveCount** - celkový počet smazaných rezervací na přístroje, cvičení a produkty v rámci dne **date**,
- **creditCount** - celkový počet odebraných kreditů v rámci dne **date**,
- **bonusCount** - celkový počet odebraných bonusů v rámci dne **date**,
- **backCredit** - celkový počet vrácených kreditů tím, že zákazník smazal rezervaci v rámci dne **date**,
- **backBonus** - celkový počet vrácených bonusů tím, že zákazník smazal rezervaci v rámci dne, **date**,
- **addCreditCount** - celkový počet vložených kreditů zákazníkům v rámci dne **date**,

- **addBonusCount** - celkový počet vložených bonusů zákazníkům v rámci dne **date**,
- **removeCreditCount** - celkový počet odebraných kreditů zákazníkům v rámci dne **date**,
- **removeBonusCount** - celkový počet odebraných bonusů zákazníkům v rámci dne **date**.

A.4.2 Tabulka statistic_pay_type

- **statisticDayCountID** - jednoznačné určení řádku tabulky,
- **date** - den ve kterém se dané rezervace prováděly,
- **resCount** - celkový počet rezervací jak přístrojů, cvičení tak produktů v rámci dne **date**,
- **resRemoveCount** - celkový počet smazaných rezervací na přístroje, cvičení a produkty v rámci dne **date**,
- **creditCount** - celkový počet odebraných kreditů v rámci dne **date**,
- **bonusCount** - celkový počet odebraných bonusů v rámci dne **date**,
- **backCredit** - celkový počet vrácených kreditů tím, že zákazník smazal rezervaci v rámci dne **date**,
- **backBonus** - celkový počet vrácených bonusů tím, že zákazník smazal rezervaci v rámci dne, **date**,
- **addCreditCount** - celkový počet vložených kreditů zákazníkům v rámci dne **date**,
- **addBonusCount** - celkový počet vložených bonusů zákazníkům v rámci dne **date**,
- **payTypeID** - tento atribut určuje o jaký typ rezervací se jedná, jestli to jsou rezervace na cvičení, přístroje, kombinované přístroje a nebo objednané produkty v obchodě,
- **PayTypeTableID** - tento atribut určuje ID daného přístroje, cvičení či produktu v rámci daného typu **payTypeID**.